

# **NAVAL POSTGRADUATE SCHOOL**

## **Monterey, California**



## **THESIS**

**SOFTWARE RE-ENGINEERING OF THE HUMAN  
FACTORS ANALYSIS AND CLASSIFICATION SYSTEM –  
(MAINTENANCE EXTENSION) USING OBJECT  
ORIENTED METHODS IN A MICROSOFT  
ENVIRONMENT**

by

Thomas P. Flanders  
and  
Scott K. Tufts

September 2001

Thesis Advisor:

Thomas Wu

Thesis Co-Advisor:

Chris Eagle

**Approved for public release; distribution is unlimited**

## Report Documentation Page

<b>Report Date</b> 30 Sep 2001	<b>Report Type</b> N/A	<b>Dates Covered (from... to)</b> -
<b>Title and Subtitle</b> Software Re- engineering of the Human Factors Analysis and Classification System (Maintenance Extension) Using Object Oriented Methods in a Microsoft Environment.		<b>Contract Number</b>
		<b>Grant Number</b>
		<b>Program Element Number</b>
<b>Author(s)</b> Flanders, Thomas P. and Tufts, Scott K.		<b>Project Number</b>
		<b>Task Number</b>
		<b>Work Unit Number</b>
<b>Performing Organization Name(s) and Address(es)</b> Research Office Naval Postgraduate School Monterey, Ca 93943-5138		<b>Performing Organization Report Number</b>
<b>Sponsoring/Monitoring Agency Name(s) and Address(es)</b>		<b>Sponsor/Monitor's Acronym(s)</b>
		<b>Sponsor/Monitor's Report Number(s)</b>
<b>Distribution/Availability Statement</b> Approved for public release, distribution unlimited		
<b>Supplementary Notes</b>		
<b>Abstract</b>		
<b>Subject Terms</b>		
<b>Report Classification</b> unclassified		<b>Classification of this page</b> unclassified
<b>Classification of Abstract</b> unclassified		<b>Limitation of Abstract</b> UU
<b>Number of Pages</b> 433		

<b>REPORT DOCUMENTATION PAGE</b>			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
<b>1. AGENCY USE ONLY (Leave blank)</b>		<b>2. REPORT DATE</b> September 2001	<b>3. REPORT TYPE AND DATES COVERED</b> Master's Thesis	
<b>4. TITLE AND SUBTITLE:</b> Software Re-engineering of the Human Factors Analysis and Classification System – (Maintenance Extension) Using Object Oriented Methods in a Microsoft Environment.			<b>5. FUNDING NUMBERS</b>	
<b>6. AUTHOR(S)</b> Flanders, Thomas P. and Tufts, Scott K.				
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Naval Postgraduate School Monterey, CA 93943-5000			<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>	
<b>9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> N/A			<b>10. SPONSORING / MONITORING AGENCY REPORT NUMBER</b>	
<b>11. SUPPLEMENTARY NOTES</b> The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
<b>12a. DISTRIBUTION / AVAILABILITY STATEMENT</b> Approved for public release; distribution is unlimited			<b>12b. DISTRIBUTION CODE</b>	
<b>13. ABSTRACT (maximum 200 words)</b> <p>The purpose of this research is to technically evaluate, refine, and expand two existing aircraft safety management information systems (one military and one civilian). The systems are used in the data collection, organization, query, analysis, and reporting of maintenance errors that contribute to Aviation mishaps, equipment damage, and personnel injury. Both programs implement the Human Factors Analysis and Classification System (HFACS) taxonomy model developed by the Naval Safety Center (NSC) to capture aircrew errors in Naval Aviation mishaps. The goal of this taxonomy is to identify areas for potential intervention by fully describing factors that are precursors to aircraft accidents.</p> <p>Requirements outlined by Dr. John K. Schmidt of the Naval Safety Center, in conjunction with funding by the National Aeronautics &amp; Space Administration, require that the system utilize a <i>Microsoft Access</i> based implementation. This research focuses on meticulous software engineering to investigate the feasibility of adapting the current "structured" systems to <i>Microsoft</i>-based object oriented architectures ensuring future scalability and increased potential for code-reuse.</p> <p>Primary research questions investigated in this thesis include: 1) How can a <i>Microsoft Access</i> based implementation provide multi-user access to the same database in a client-server environment while ensuring the ability to scale to a large number (potentially thousands) of users? 2) How can the linguistic discontinuity associated with object-oriented concepts and non-object oriented, flat relational databases be overcome when limited by the requirement for a <i>Microsoft Access</i> based solution? 3) The current military and civilian systems provide similar functionality, but use different database schema. How can object oriented methods be implemented to provide a common interface to both types of data? 4) How should database schema be changed to provide the best performance, scalability, and opportunity for code re-use? 5) In the past, <i>Microsoft</i> has deployed new versions of <i>Microsoft Access</i> and <i>Visual Basic</i> that were not (fully) backwards compatible with previous versions. This caused great discontent among users of applications designed to run under the older versions of these programs. How can our system(s) be designed to isolate them from problems associated with new versions of <i>Microsoft</i> products? Specifically, the pending release of <i>Microsoft Office 2002</i>, the new <i>SQL Server 2000</i> database engine, and <i>Microsoft Visual Basic.NET</i>.</p> <p>This thesis describes our use of the Spiral Development Model to create a <i>Microsoft</i> Based solution for the Aviation Safety School requirements. We hypothesize that this research produced products that greatly enhance current HFACS-capabilities and provide the means to weather further changes in requirements and application platforms.</p>				
<b>14. SUBJECT TERMS</b> Aviation Safety, Microsoft Access, HFACS, HFACS-ME, Flanders, Tufts, FAA, Federal Aviation Administration, NASA, National Aeronautics and Space Administration			<b>15. NUMBER OF PAGES</b>	
			<b>16. PRICE CODE</b>	
<b>17. SECURITY CLASSIFICATION OF REPORT</b> Unclassified	<b>18. SECURITY CLASSIFICATION OF THIS PAGE</b> Unclassified	<b>19. SECURITY CLASSIFICATION OF ABSTRACT</b> Unclassified	<b>20. LIMITATION OF ABSTRACT</b> UL	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

**SOFTWARE RE-ENGINEERING OF THE HUMAN FACTORS ANALYSIS AND  
CLASSIFICATION SYSTEM – (MAINTENANCE EXTENSION) USING  
OBJECT ORIENTED METHODS IN A MICROSOFT ENVIRONMENT**

Thomas P. Flanders  
Major, United States Army  
B.S., Clarkson University, 1989  
M.A., Webster University, 2000

Scott K. Tufts  
Major, United States Army  
B.S., United States Military Academy, 1990

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

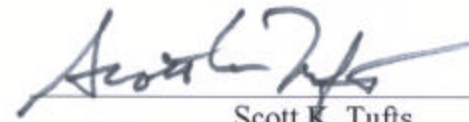
**NAVAL POSTGRADUATE SCHOOL  
September 2001**

Author:



Thomas P. Flanders

Author:



Scott K. Tufts

Approved by:



Thomas Wu, Thesis Advisor



Chris Eagle, Thesis Co-Advisor



Chris Eagle, Chairman  
Computer Science Department

THIS PAGE INTENTIONALLY LEFT BLANK

## ABSTRACT

The purpose of this research is to technically evaluate, refine, and expand two existing aircraft safety management information systems (one military and one civilian). The systems are used in the data collection, organization, query, analysis, and reporting of maintenance errors that contribute to Aviation mishaps, equipment damage, and personnel injury. Both programs implement the Human Factors Analysis and Classification System (HFACS) taxonomy model developed by the Naval Safety Center (NSC) to capture aircrew errors in Naval Aviation mishaps. The goal of this taxonomy is to identify areas for potential intervention by fully describing factors that are precursors to aircraft accidents.

Requirements outlined by Dr. John K. Schmidt of the Naval Safety Center, in conjunction with funding by the National Aeronautics & Space Administration, require that the system utilize a *Microsoft Access* based implementation. This research focuses on meticulous software engineering to investigate the feasibility of adapting the current "structured" systems to *Microsoft*-based object oriented architectures ensuring future scalability and increased potential for code-reuse.

Primary research questions investigated in this thesis include: 1) How can a *Microsoft Access* based implementation provide multi-user access to the same database in a client-server environment while ensuring the ability to scale to a large number (potentially thousands) of users? 2) How can the linguistic discontinuity associated with object-oriented concepts and non-object oriented, flat relational databases be overcome when limited by the requirement for a *Microsoft Access* based solution? This problem is commonly called "impedance Mismatch". 3) The current military and civilian systems provide similar functionality, but use different database schema. How can object oriented methods be implemented to provide a common interface to both types of data? 4) How should database schema be changed to provide the best performance, scalability, and opportunity for code re-use? 5) In the past, *Microsoft* has deployed new versions of *Microsoft Access* and *Visual Basic* that were not (fully) backwards compatible with previous versions. This caused great discontent among users of applications designed to

run under the older versions of these programs. How can our system(s) be designed to isolate them from problems associated with new versions of *Microsoft* products? Specifically, the pending release of *Microsoft Office 2002*, the new *SQL Server 2000* database engine, and *Microsoft Visual Basic.NET*.

This thesis describes our use of the Spiral Development Model to create a *Microsoft* Based solution for the Aviation Safety School requirements. We hypothesize that the prototype produced as a part of our research will greatly enhance current HFACS-capabilities and provide the means to weather further changes in requirements and application platforms.



## TABLE OF CONTENTS

<b>I.</b>	<b>INTRODUCTION.....</b>	<b>1</b>
<b>A.</b>	<b>BACKGROUND .....</b>	<b>1</b>
<b>B.</b>	<b>AREA OF RESEARCH/SCOPE .....</b>	<b>5</b>
<b>C.</b>	<b>REQUIREMENTS .....</b>	<b>6</b>
<b>D.</b>	<b>METHODOLOGY .....</b>	<b>11</b>
	1. Phase I - Requirements Analysis .....	11
	2. Phase II - System Foundation Development/Implementation.....	12
	3. Phase III - HFACS-ME Development/Implementation .....	12
	4. Phase IV - Test and Analysis.....	12
<b>E.</b>	<b>ASSUMPTIONS.....</b>	<b>12</b>
<b>F.</b>	<b>DEFINITIONS .....</b>	<b>12</b>
<b>G.</b>	<b>ORGANIZATION .....</b>	<b>13</b>
<b>II.</b>	<b>REQUIREMENTS ANALYSIS .....</b>	<b>15</b>
<b>A.</b>	<b>OVERVIEW.....</b>	<b>15</b>
<b>B.</b>	<b>USE CASE ANALYSIS .....</b>	<b>15</b>
	1. Query Database.....	17
	a. <i>Query by Single Field.....</i>	<i>17</i>
	b. <i>Query by Multiple Fields .....</i>	<i>18</i>
	c. <i>Create a Report.....</i>	<i>18</i>
	d. <i>HFACS-ME Summary.....</i>	<i>19</i>
	e. <i>Create a Graph.....</i>	<i>19</i>
	2. Add to Database.....	20
	a. <i>Add a mishap.....</i>	<i>20</i>
	b. <i>Add Factor.....</i>	<i>21</i>
	3. Edit Records in Database.....	21
	a. <i>Edit Mishap .....</i>	<i>22</i>
	b. <i>Edit Factor.....</i>	<i>23</i>
	4. Change Server .....	23
	5. Replace the Database.....	24
	a. <i>Replace the Database via FTP.....</i>	<i>25</i>
	b. <i>Replace the Database via Disk.....</i>	<i>25</i>
<b>C.</b>	<b>CLASS-RESPONSIBILITY-COLLABORATION (CRC) CARDS.....</b>	<b>26</b>
<b>D.</b>	<b>MICROSOFT ACCESS &amp; DATABASE ENGINES .....</b>	<b>27</b>
<b>E.</b>	<b>DATA ACCESS TECHNOLOGIES .....</b>	<b>28</b>
	1. OLE DB.....	29
	2. ADO.....	30
	3. ODBC.....	31
	4. DAO.....	31
	5. RDO.....	31
	6. SQLDMO.....	32

F.	PROGRAMMING MICROSOFT ACCESS AND SQL SERVER .....	33
G.	MICROSOFT DEVELOPMENT EFFORTS .....	35
1.	Access 2002 [Ref. 16].....	35
2.	Visual Basic.NET [Ref. 17].....	36
3.	SQL Server .....	37
H.	THE CONCEPTUAL MODEL .....	37
III.	HFACS CONNECTIVITY COMPONENT DEVELOPMENT.....	41
A.	OVERVIEW .....	41
B.	SEQUENCE DIAGRAMS.....	41
1.	Change Server .....	42
2.	Replace the Database via FTP .....	42
3.	Replace the Database via Disk.....	43
C.	COLLABORATION DIAGRAMS .....	43
1.	Change Server .....	43
2.	Replace the Database via FTP .....	44
3.	Replace the Database via Disk.....	44
D.	CLASS DIAGRAMS .....	44
1.	HFACS Connection Class .....	45
2.	HFACS_Main Class.....	46
3.	UpdateController Class .....	46
4.	UpdateDisk Class .....	47
5.	FTPUpdate Class .....	47
6.	MSDE Class.....	47
E.	IDENTIFICATION OF SDM STAGES .....	48
F.	IMPLEMENTATION - STAGE 1 .....	49
G.	IMPLEMENTATION - STAGE 2 .....	55
H.	IMPLEMENTATION - STAGE 3 .....	58
I.	SUMMARY.....	61
IV.	HFACS BUSINESS COMPONENT DEVELOPMENT.....	63
A.	OVERVIEW .....	63
B.	ARCHITECTURE.....	63
C.	SEQUENCE DIAGRAMS.....	66
1.	Add Factors .....	67
2.	Add Mishap .....	68
3.	Graph .....	69
4.	Edit a Mishap .....	69
5.	Edit a Factor.....	70
6.	Get Summary Report .....	71
7.	Create a Report .....	71
8.	Query .....	72
D.	COLLABORATION DIAGRAMS .....	72
1.	Add Factors .....	73
2.	Add Mishaps .....	73
3.	Graph .....	74
4.	Edit a Mishap .....	74

5.	Edit a Factor.....	74
6.	Get Summary Report .....	75
7.	Create a Report .....	75
8.	Query .....	76
E.	CLASS DIAGRAMS .....	76
1.	Main Menu Class .....	78
2.	Connection Functions Class.....	78
3.	Select Mishap Class.....	79
4.	Edit Mishap Class .....	79
5.	Add Mishap Class .....	79
6.	Expert Graph Class .....	80
7.	Actual Graph Class.....	80
8.	Query Menu Class.....	81
9.	Summary Class.....	81
10.	Expert Query Class.....	82
11.	View Mishaps Class .....	82
12.	Report Class.....	82
F.	IDENTIFICATION OF SDM STAGES .....	83
G.	IMPLEMENTATION - STAGE 1 .....	83
H.	IMPLEMENTATION - STAGE 2 .....	86
I.	IMPLEMENTATION - STAGE 3 .....	88
J.	IMPLEMENTATION - STAGE 4 .....	90
K.	IMPLEMENTATION - STAGE 5 .....	92
1.	Windows 98 Tests.....	92
2.	Windows 2000 Tests.....	93
V.	CONCLUSIONS AND RECOMMENDATIONS.....	95
A.	CONCLUSIONS .....	95
B.	RECOMMENDATIONS.....	100
C.	SUMMARY.....	101
	APPENDIX A. CRC CARDS DEVELOPED FOR HFASC-ME.....	103
A.	CONNECTION COMPONENT CRC CARDS .....	103
B.	BUSINESS LOGIC COMPONENTS CRC CARDS.....	105
	APPENDIX B. CLASS DIAGRAMS.....	111
A.	FACS.DLL CLASS DIAGRAM .....	111
B.	HFACSFTP.EXE CLASS DIAGRAM .....	111
	APPENDIX C. DESCRIPTION OF CLASSES .....	113
A.	HFACS CONNECTION CLASS.....	113
1.	Class Description.....	113
2.	Data Member Description.....	113
3.	Method Description.....	114
B.	ODBLOGON CLASS .....	115
1.	Class Description.....	115
2.	Data Member Description.....	115
3.	Method Description.....	116

C.	UPDATECONTROLLER CLASS .....	116
1.	Class Description.....	116
2.	Data Member Description.....	116
3.	Method Description.....	117
D.	DISK UPDATE CLASS .....	117
1.	Class Description.....	117
2.	Data Member Description.....	117
3.	Method Description.....	117
E.	FTPUPDATE CLASS.....	118
1.	Class Description.....	118
2.	Data Member Description.....	118
3.	Method Description.....	118
F.	MSDE CLASS .....	119
1.	Class Description.....	120
2.	Data Member Description.....	120
3.	Method Description.....	120
G.	CALLBACK CLASS .....	122
1.	Class Description.....	122
2.	Data Member Description.....	122
3.	Method Description.....	123
H.	INIFILE CLASS .....	123
1.	Class Description.....	123
2.	Data Member Description.....	123
3.	Method Description.....	123
I.	HFACSMAN CLASS .....	124
1.	Class Description.....	124
2.	Data Member Description.....	125
J.	INIFILECONTROLLER CLASS .....	127
1.	Class Description.....	127
2.	Data Member Description.....	127
3.	Method Description.....	127
K.	WAIT CLASS.....	128
1.	Class Description.....	128
1.	Data Member Description.....	128
2.	Method Description.....	128
L.	WELCOME CLASS .....	128
1.	Class Description.....	128
2.	Data Member Description.....	129
3.	Method Description.....	129
M.	CONSTRUCTORS CLASS .....	129
1.	Class Description.....	129
2.	Data Member Description.....	129
3.	Method Description.....	129
N.	ERRORLOG CLASS .....	130
1.	Class Description.....	130

	2.	Data Member Description.....	130
	3.	Method Description.....	130
O.		FTPCBK CLASS .....	130
	1.	Class Description.....	131
	2.	Data Member Description.....	131
	3.	Method Description.....	131
P.		TIMER CLASS .....	131
	1.	Class Description.....	131
	2.	Data Member Description.....	131
	3.	Method Description.....	131
Q.		FTP CLASS .....	132
	1.	Class Description.....	132
	2.	Data Member Description.....	132
	3.	Method Description.....	134
APPENDIX D.....			137
APPENDIX E. DESCRIPTION OF BUSINESS LOGIC CLASSES .....			139
A.		INIFILE CLASS .....	139
	1.	Class Description.....	139
	2.	Data Member Description.....	139
	3.	Method Description.....	139
B.		GLOBALDECLARATIONS CLASS .....	140
	1.	Class Description.....	140
	2.	Data Member Description.....	140
	3.	Method Description.....	140
C.		DETERMINEOSDECLARES CLASS .....	141
	1.	Class Description.....	141
	2.	Data Member Description.....	141
	3.	Method Description.....	142
D.		FORMWINDOW CLASS .....	143
	1.	Class Description.....	143
	2.	Data Member Description.....	143
	3.	Method Description.....	144
E.		SIZING FUNCTION CLASS .....	145
	1.	Class Description.....	145
	2.	Data Member Description.....	145
	3.	Method Description.....	145
F.		SELECT MISHAP CLASS .....	146
	1.	Class Description.....	146
	2.	Data Member Description.....	147
	3.	Method Description.....	147
G.		SUB SELECT MISHAP CLASS .....	147
	1.	Class Description.....	148
	2.	Data Member Description.....	148
	3.	Method Description.....	148
H.		EDIT MISHAP CLASS .....	148

	1. Class Description.....	149
	2. Data Member Description.....	149
	3. Method Description.....	149
I.	MISHAP DESCRIPTION CLASS .....	150
	1. Class Description.....	150
	2. Data Member Description.....	150
	3. Method Description.....	150
J.	FACTORS CLASS.....	151
	1. Class Description.....	151
	2. Data Member Description.....	151
	3. Method Description.....	151
K.	ADD MISHAP CLASS .....	152
	1. Class Description.....	152
	2. Data Member Description.....	152
	3. Method Description.....	152
L.	CODE MAINTENANCE CLASS .....	154
	1. Class Description.....	155
	2. Data Member Description.....	155
	3. Method Description.....	155
M.	CLOSE COMMAND CLASS .....	155
	1. Class Description.....	155
	2. Data Member Description.....	155
	3. Method Description.....	155
N.	CONNECTION FUNCTIONS CLASS.....	156
	1. Class Description.....	156
	2. Data Member Description.....	156
	3. Method Description.....	156
O.	PLEASE WAIT CLASS .....	158
	1. Class Description.....	158
	2. Data Member Description.....	158
	3. Method Description.....	158
P.	MAIN MENU CLASS .....	159
	1. Class Description.....	159
	2. Data Member Description.....	159
	3. Method Description.....	159
Q.	ACTUAL GRAPH CLASS .....	161
	1. Class Description.....	161
	2. Data Member Description.....	161
	3. Method Description.....	162
R.	EXPERT GRAPH CLASS .....	163
	1. Class Description.....	164
	2. Data Member Description.....	164
	3. Method Description.....	164
S.	SUMMARY CLASS .....	165
	1. Class Description.....	165

	2.	Data Member Description.....	166
	3.	Method Description.....	166
T.		VIEW MISHAPS CLASS .....	168
	1.	Class Description.....	168
	2.	Data Member Description.....	168
	3.	Method Description.....	168
U.		EXPERT QUERY CLASS .....	169
	1.	Class Description.....	169
	2.	Data Member Description.....	170
	3.	Method Description.....	170
V.		QUERY MENU CLASS .....	171
	1.	Class Description.....	172
	2.	Data Member Description.....	172
	3.	Method Description.....	172
W.		REPORT CLASS .....	173
	1.	Class Description.....	173
	2.	Data Member Description.....	173
	3.	Method Description.....	173
APPENDIX F. BUSINESS LOGIC COMPONENT CODE .....			175
APPENDIX G. CONNECTION COMPONENT .....			269
APPENDIX H. CLIPBOARD UTILITY .....			313
APPENDIX I. FTP SERVER.....			315
APPENDIX J. INSTALL CD CODE .....			323
APPENDIX K. INVESTIGATION MODULE .....			327
APPENDIX L. MODIFIED VB SETUP1 .....			363
APPENDIX M. STORED PROCEDURES .....			367
LIST OF REFERENCES .....			405
INITIAL DISTRIBUTION LIST .....			409

THIS PAGE INTENTIONALLY LEFT BLANK



## LIST OF FIGURES

Figure 1.1a.	Management Conditions Category. ....	2
Figure 1.1b.	Working Conditions Category. ....	3
Figure 1.1c.	Maintainer Conditions Category. ....	3
Figure 1.1d.	Maintainer Acts Categories. ....	4
Figure 1.2.	Example HFACS Summary. ....	9
Figure 1.3.	Example HFACS Report Output. ....	10
Figure 2.1.	HFACS-ME Use Cases (1 <sup>st</sup> Level). ....	16
Figure 2.2.	Query Database Use Case. ....	17
Figure 2.3.	Add to Database Use Case. ....	20
Figure 2.4.	Edit a Record in Database Use Case. ....	21
Figure 2.5.	Replace the Database Use Case. ....	24
Figure 2.6.	OLE DB Architecture. ....	30
Figure 2.7.	Conceptual Model for the Connection Component. ....	38
Figure 2.8.	Conceptual Model for the Business -Logic Component. ....	39
Figure 2.9.	Conceptual Architecture at the End of Requirements Analysis. ....	40
Figure 3.1.	Change Server Sequence Diagram. ....	42
Figure 3.2.	Replace the Database via FTP Sequence Diagram. ....	42
Figure 3.3.	Replace the Database via Disk Sequence Diagram. ....	43
Figure 3.4.	Change Server Collaboration Diagram. ....	43
Figure 3.5.	Replace the Database via FTP Collaboration Diagram. ....	44
Figure 3.6.	Replace the Database via FTP Collaboration Diagram. ....	44
Figure 3.7.	Interim Class Diagram. ....	45
Figure 3.8.	Class Diagram for HFACS Connection. ....	46
Figure 3.9.	HFACS_Main Class Diagram. ....	46
Figure 3.10.	Class Diagram for UpdateController Class. ....	46
Figure 3.11.	Class Diagram for UpdateDisk Class. ....	47
Figure 3.12.	Class Diagram for FTPUpdate Class. ....	47
Figure 3.13.	Class Diagram for MSDE Class. ....	48
Figure 3.14.	OLE DB Architecture. ....	55
Figure 3.15.	File Install Locations. ....	58
Figure 4.1.	Add Factor Sequence Diagram. ....	67
Figure 4.2.	Add Mishap Sequence Diagram. ....	68
Figure 4.3.	Graph Sequence Diagram. ....	69
Figure 4.4.	Edit a Mishap Sequence Diagram. ....	69
Figure 4.5.	Edit a Factor Sequence Diagram. ....	70
Figure 4.6.	Summary Report Sequence Diagram. ....	71
Figure 4.7.	Create a Report Sequence Diagram. ....	71
Figure 4.8.	Query Sequence Diagram. ....	72
Figure 4.9.	Add Factors Collaboration Diagram. ....	73
Figure 4.10.	Add Mishaps Collaboration Diagram. ....	73
Figure 4.11.	Graph Collaboration Diagram. ....	74

Figure 4.12.	Edit a Mishap Collaboration Diagram. ....	74
Figure 4.13.	Edit a Factor Collaboration Diagram. ....	74
Figure 4.14.	Get Summary Report Collaboration Diagram. ....	75
Figure 4.15.	Create a Report Collaboration Diagram. ....	75
Figure 4.16.	Query Collaboration Diagram. ....	76
Figure 4.17.	Intermediate Class Diagram. ....	77
Figure 4.18.	Main Menu Class Diagram. ....	78
Figure 4.19.	Connection Functions Class Diagram. ....	78
Figure 4.20.	Class Diagram for Select Mishap Class. ....	79
Figure 4.21.	Edit Mishap Class Diagram. ....	79
Figure 4.22.	Add Mishap Class Diagram. ....	79
Figure 4.23.	Expert Graph Class Diagram. ....	80
Figure 4.24.	Actual Graph Class Diagram. ....	80
Figure 4.25.	Query Menu Class Diagram. ....	81
Figure 4.26.	Summary Class Diagram. ....	81
Figure 4.27.	Expert Query Class Diagram. ....	82
Figure 4.28.	View Mishaps Class Diagram. ....	82
Figure 4.29.	Report Class Diagram. ....	82
Figure 4.30.	HFACS Tables - 3rd Normal Form. ....	85
Figure 4.31.	HFACS Tables - Final Solution. ....	86
Figure 4.32.	Example Crosstab Query. ....	91
Figure C.1.	Class Diagram for HFACS Connection. ....	113
Figure C.2.	Class Diagram for ODBLogon. ....	115
Figure C.3.	Class Diagram for UpdateController Class. ....	116
Figure C.4.	Class Diagram for Disk Update Class. ....	117
Figure C.5.	Class Diagram for FTPUpdate Class. ....	118
Figure C.6.	Class Diagram for MSDE Class. ....	119
Figure C.7.	Class Diagram for Callback Class. ....	122
Figure C.8.	INIFile Class Diagram. ....	123
Figure C.9.	HFACSMain Class Diagram. ....	124
Figure C.10.	INIFileController Class Diagram. ....	127
Figure C.11.	Wait Class Diagram. ....	128
Figure C.12.	Welcome Class Diagram. ....	128
Figure C.13.	Constructors Class Diagram. ....	129
Figure C.14.	ErrorLog Class Diagram. ....	130
Figure C.15.	FTPCBK Class Diagram. ....	130
Figure C.16.	Timer Class Diagram. ....	131
Figure C.17.	FTP Class Diagram. ....	132
Figure E.1.	Class Diagram for INIFile Class. ....	139
Figure E.2.	Class Diagram for GlobalDeclaration Class. ....	140
Figure E.3.	Class Diagram for DetermineOSDeclares Class. ....	141
Figure E.4.	Class Diagram for FormWindow Class. ....	143
Figure E.5.	Class Diagram for Sizing Function Class. ....	145
Figure E.6.	Class Diagram for Select Mishap Class. ....	146
Figure E.7.	Class Diagram for Sub Select Mishap Class. ....	147

Figure E.8.	Edit Mishap Class Diagram. ....	148
Figure E.9.	Mishap Description Class Diagram. ....	150
Figure E.10.	Factors Class Diagram. ....	151
Figure E.11.	Add Mishap Class Diagram. ....	152
Figure E.12.	Code Maintenance Class Diagram. ....	154
Figure E.13.	Close Command Class Diagram. ....	155
Figure E.14.	Connection Functions Class Diagram. ....	156
Figure E.15.	Please Wait Class Diagram. ....	158
Figure E.16.	Main Menu Class Diagram. ....	159
Figure E.17.	Actual Graph Class Diagram. ....	161
Figure E.18.	Expert Graph Class Diagram. ....	163
Figure E.19.	Summary Class Diagram. ....	165
Figure E.20.	View Mishaps Class Diagram. ....	168
Figure E.21.	Expert Query Class Diagram. ....	169
Figure E.22.	Query Menu Class Diagram. ....	171
Figure E.23.	Report Class Diagram. ....	173

THIS PAGE INTENTIONALLY LEFT BLANK

## ACKNOWLEDGMENTS

We would like to acknowledge and thank the following people for their part in the successful completion of this thesis:

Our advisors, Dr. Thomas Wu and LtCdr Chris Eagle for providing much needed advice, support, expertise, and patience.

Capt John K. Schmidt and Capt(R) George Zolla for their guidance and direction of the HFACS effort.

Brian Steckler and his team of programmers at of Universal Internet for their assistance on various aspects of the design.

Tabitha Barham, James Carr, and Keith Wong of Microsoft Corporation for their assistance in troubleshooting issues related to the SQLDMO object model and with the *Microsoft Office Developer* version of the *Package & Deployment* program.

CPT Dwight Hunt of TRAC Monterey, for his support and provision of licensed development tools.

Cdr Anthony Boex and CPT Doug Nelson for their assistance with the actual code effort.

Mr. Steve Dassin, inventor of the *Replacement for Access Crosstab* package of tools. This project would have no graphs or reports without his assistance.

Mr. Andy Irvine of Explain Limited, who provided greatly needed assistance troubleshooting advanced SQL queries.

Our wives, Kelley Flanders and MiKyong Tufts for their unwavering support.

THIS PAGE INTENTIONALLY LEFT BLANK

## EXECUTIVE SUMMARY

The purpose of this research is to technically evaluate, refine, and expand two existing aircraft safety management information systems (one military and one civilian). The systems are used in the data collection, organization, query, analysis, and reporting of maintenance errors that contribute to Aviation mishaps, equipment damage, and personnel injury. Both programs implement the Human Factors Analysis and Classification System (HFACS) taxonomy model developed by the Naval Safety Center (NSC) to capture aircrew errors in Naval Aviation mishaps. The goal of this taxonomy is to identify areas for potential intervention by fully describing factors that are precursors to aircraft accidents.

Requirements outlined by Dr. John K. Schmidt of the Naval Safety Center, in conjunction with funding by the National Aeronautics & Space Administration, require that the system utilize a *Microsoft Access* based implementation. This research focuses on meticulous software engineering to investigate the feasibility of adapting the current "structured" systems to *Microsoft*-based object oriented architectures ensuring future scalability and increased potential for code-reuse.

THIS PAGE INTENTIONALLY LEFT BLANK



# **I. INTRODUCTION**

## **A. BACKGROUND**

The Human Factors Analysis Classification System - Maintenance Extension (HFACS-ME) is a tool used in the data collection, organization, query, analysis, and reporting of maintenance errors that contribute to Aviation mishaps, equipment damage, and personnel injury. In order to better relate the scope and requirements of the software reengineering and development efforts outlined in this thesis, a general overview of the Human Factors Analysis model is in order.

Aircraft accidents occur due to many contributing factors. No matter how obvious the cause of an accident may appear, an investigation is always performed after the fact to ensure that *all* underlying causes for the mishap are captured. Great emphasis is placed on the word *all*. A failure to fully describe the causes of a mishap can result in oversights that allow future mishaps of the same type to occur. Research related to accident investigations has demonstrated that in seventy to eighty percent of civil and military aircraft accidents, the underlying causes were human errors [Ref. 25]. Furthermore, close to ninety-two percent of investigations into Naval Reserve Aviation mishaps cited maintenance personnel as the primary causal factor for the mishap [Ref. 1].

The Naval Safety Center (NSC) recognized the need to develop a formal process for categorizing the causes of aviation mishaps in an attempt to prevent them from recurring. In response, it developed a Human Factors Analysis and Classification System (HFACS) taxonomy. The HFACS model incorporates Reason's model of latent and active failures [Ref. 27] as well as Heinrich's "Domino Theory" [Ref. 28] and Edward's "SHEL model" [Ref. 29]. In general, the model facilitates classification of errors and violations associated with a mishap into several broad categories. Once categorized, the mishap data is much easier to manipulate and analyze, enhancing problem solving techniques. Examples of categories in the original model include:

- Supervisory conditions. Inadequate supervision, planning inappropriate tasks, failure to correct known problems, and supervisory violations.

- Operator conditions. Adverse physical and mental states, which include situational awareness, mental fatigue, over confidence, complacency, visual illusions, hypoxia, poor communication, not assertive, intoxication, mental lapses, and illness.
- Workplace conditions. Confining space, damaged equipment, using uncertified equipment, inadequate lighting, adverse weather, and inaccessible workspace.

In 1995, the NSC officially adopted the HFACS model as the standard for analyzing human errors in Naval Aviation mishaps and targeting appropriate prevention. Although there was some reduction in the Naval aviation mishap rate with the implementation of HFACS, its restricted focus on only aircrew errors limited its utility. A 1997 study by Schmidt, J., Schmorow, D., & Hardee, M. noted that HFACS could be extended to cover maintenance errors [Ref. 26]. As a result of this study, a Maintenance Extension (ME) of the HFACS taxonomy was adapted to classify causal factors that contributed to maintenance mishaps. The additions to the model focused on detailing how latent factors that contribute to a maintainer's performance could possibly lead to an active failure or ultimately an unsafe maintainer act. The new model (depicted in Figures 1-1a,b,c,d) consists of four major categories each broken down into three levels of inter-related factors. This new taxonomy can truly be used to define all possible mishap related factors.

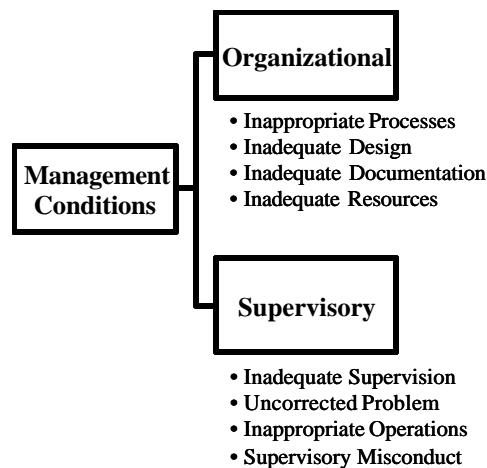


Figure 1.1a. Management Conditions Category.

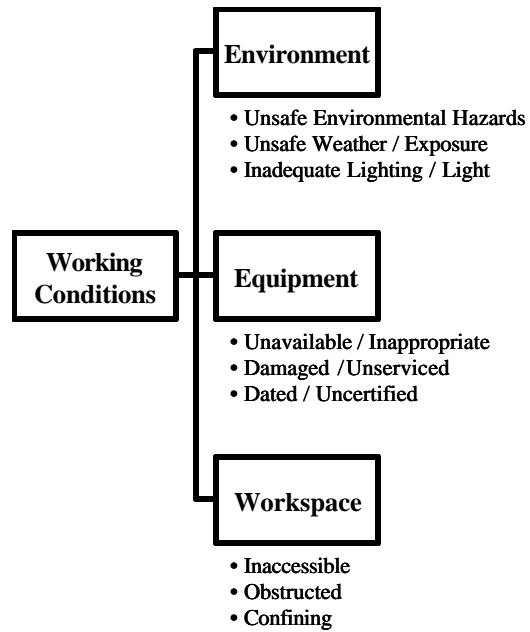


Figure 1.1b. Working Conditions Category.

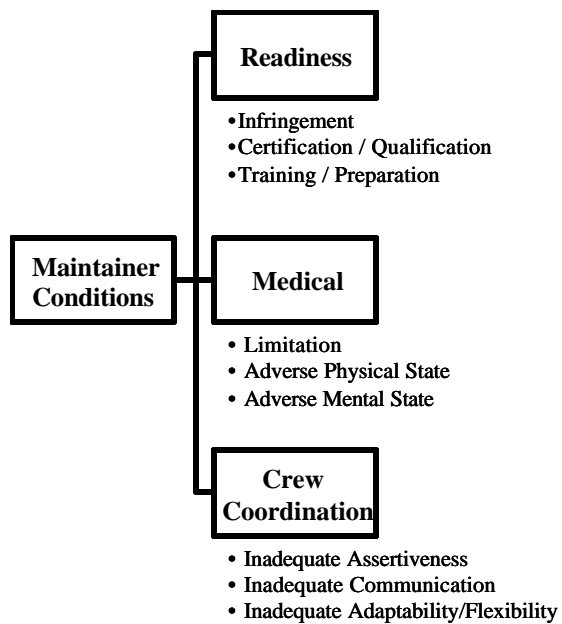


Figure 1.1c. Maintainer Conditions Category.

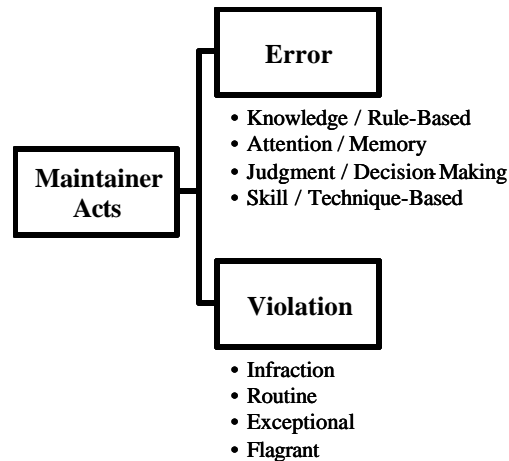


Figure 1.1d. Maintainer Acts Categories.

A 1998 review of 470 Naval Aviation mishaps determined that the new HFACS-ME taxonomy was indeed an effective classification system for determining trends in aviation mishaps [Ref. 33]. Building on Schmorow's research, Fry developed the first partially automated HFACS-ME model implementation [Ref. 34]. Dubbed the "Maintenance Error Information Management System" (MEIMS), the new tool effectively could handle more data than its paper-based predecessor -- refining the HFACS-ME model and making it more efficient and effective. Over time, more automated improvements were desired. Fry's rudimentary MIEMS spreadsheet-based tool was further refined by Wood and developed into a working prototype stand-alone application [Ref. 2]. This new *Microsoft Access 97* based program was distributed for Fleet testing and evaluation. A follow-on usability study of the prototype determined that it could be developed into an effective system, not only in determining trends but providing information for mishap prevention efforts.

In the period between June 2000 and January 2001, the HFACS-ME *Microsoft Access 97* database underwent various modifications to enhance its capabilities and make it compatible with *Microsoft Access 2000*. A civilian variant was developed using a different set of database schema in order to investigate application of the HFACS-ME model to the commercial aviation industry. In January 2001, Dr. John Schmidt demonstrated this prototype civilian HFACS-ME system to representatives from NASA and the FAA. As a result of this meeting, NASA provided funding to support the

development of entirely new prototypes for both the civilian and military versions of HFACS-ME.

This thesis is part of the new HFACS-ME prototype development effort. There are four separate groups working on different areas of the project:

- Group 1- Responsible for this thesis, encompassing the software engineering and implementation of desktop prototypes for both military and civilian versions of HFACS-ME.
- Group 2 - Responsible for web-enabling the database with support from an external contractor.
- Group 3 - Responsible for refinements in the existing military and civilian versions of HFACS-ME and developing requirements for groups 1 and 2. Also responsible for an independent usability study of our redesigned HFACS programs.
- Group 4 - Responsible for developing a distance learning interface for the entire system.

## **B. AREA OF RESEARCH/SCOPE**

A well-designed HFACS-ME information management system capable of weathering upgrades to platform applications while providing scalability and opportunity for code reuse will ensure the satisfaction of its users for many years. Providing a user-friendly interface to the application will ensure standardization of data input and increase the validity and reliability of the data for investigators and safety personnel. Access to this data will allow maintainers and safety personnel to quickly identify potential hazards, analyze trends and ultimately train personnel to avoid future occurrences, reducing aircraft mishaps and potentially saving lives.

This thesis is part of ongoing effort to investigate the feasibility of the HFACS-ME as a taxonomy framework for the investigation, collection, and analysis of maintenance related mishap data with the use of the MEIMS. Our research will enable us to further refine both versions of HFACS-ME in conjunction with the NASA requirements and other groups working on their respective areas of the project. The specific questions we will attempt to answer are:

- How can a *Microsoft Access* based implementation provide multi-user access to the same database in a client-server environment while ensuring the ability to scale to a large number (potentially thousands) of users?

- How can the linguistic discontinuity associated with object-oriented concepts and non-object oriented, flat relational databases be overcome when limited by requirements to use certain types of software implementations (e.g. a *Microsoft Access* based solution)?
- The current military and civilian systems provide similar functionality, but use different database schema. How can a common interface be developed for both types of data?
- How should database schema be changed to provide the best performance, scalability, and opportunity for code re-use?
- In the past, *Microsoft* has deployed new versions of *Microsoft Access* and *Visual Basic* that were not (fully) backwards compatible with previous versions. This caused great discontent among users of applications designed to run under the older versions of these programs. How can our systems be designed to isolate them from problems associated with new versions of *Microsoft Access*? Specifically, the pending release of *Microsoft Office XP*, *Microsoft Office 2002* and *Microsoft Visual Basic.NET*?
- What new features should be implemented to make the information systems more user interactive and user friendly?

### C. REQUIREMENTS

The purpose of this section is to identify and document requirements for the new HFACS-ME prototype in a form that clearly communicates the intent of our sponsors. We recognize the importance of correct and thorough requirements specification as one of the most important parts of this design effort. The detailed specifications herein were provided by Dr. John Schmidt of the Navy Aviation Safety Center. These requirements were established to provide enough information regarding the system to allow us to begin contemplating the conceptual model for the software engineering effort.

The primary goal of creating a desktop version of HFACS-ME is to provide a capability for investigating aviation mishaps using an efficient automated tool from a field location without network/Internet connectivity. The system should provide an intuitive graphical user interface encompassing all the functionality of the current HFACS system. In addition, it should be designed so as to provide the capability to scale into an enterprise level networked & web-enabled application. It must be adequately documented and provide maximum opportunity for code reuse. In order to facilitate rapid application development methods the system must be implemented using *Microsoft*

*Access 2000*. It must be capable of running on all *Intel* X86 (or compatible) platforms running a *Microsoft Windows 95* or newer operating system. Finally, to the maximum extent possible, the system should be developed to insulate it from compatibility problems associated with upgrades in operating systems, programming languages, and versions of *Access*.

The HFACS-ME system must be compatible with many different types of hardware ranging from notebook computers to large enterprise servers. Although the system does not have to process data in real-time, it should provide an "adequate" level of usability with the following minimum hardware specifications:

- Computer CPU: *Intel®* or compatible *Pentium* 166 MHz or higher.
- Memory (RAM): 32 MB minimum on all other operating systems
- Hard Disk Space: 75 MB minimum, 150 MB typical
- Monitor: 800x600 or higher resolution required
- Pointing Device: *Microsoft* Mouse or compatible
- CD-ROM Drive: Required
- Internet Software: *Microsoft Internet Explorer 5.0*

Two versions of the program are required, one for civilian use and the second for military use. Specific requirements for the civilian version are not well defined and are expected to grow after initial release of the program. Care should be taken to provide as much opportunity for code reuse in this area as possible. As a minimum, the following system functions and attributes must be implemented in both versions of the program:

A Main Menu. The Main Menu must have the following user options.

- Query
- HFACS-ME Summary
- Graphs
- Reports
- Add/Edit Mishap
- Exit

Details of the "Query" Option. The Query option will provide methods to search and analyze the accident database. It must allow users to query the database based on

different kinds of criteria in order to locate instances of certain types of mishaps. For each query result, the screen output should list all contributing factors associated with the mishap. This includes a description of the factor and the associated first, second and third level causal codes. There should be an option to display the HFAC-ME taxonomy so that these causal codes can be better understood. The user should be able to view one mishap at a time or display the total number of mishaps returned by the users query. There must be an option to display an expanded description of the mishap. Finally, the user must be able to query the database by selecting one or any combination of the following mishap criteria:

- Aircraft Model
- Aircraft Type
- Organization
- Location
- Mishap Class
- Mishap Type
- Year

Details of the "HFACS-ME Summary" Option. The program must offer an option to tabulate summary statistics of HFACS data that provide the user with the percentages of all HFACS-ME error categories within a group of selected accidents. This will be a mathematically intensive operation. The selection categories should be comprised of the same options as used by the Query option, as well as, all three HFACS-ME Error Category levels.

The screen output for this implementation should graphically display the HFACS-ME factors structure. It should illustrate summary statistics for each category. At a minimum the summary statistics should include number of factors and percentage of mishaps that with factor. The "Level" categories must allow the user to search the database for factors that only apply to that level. For example, the user should be able to identify which accidents involved a Maintainer Act-Violation-Infraction or a Management Condition-Supervisory-Supervisory Misconduct causal factor. This will



allow users to better identify contributing factors because the corresponding percentages of the other Error Categories will also be visible on one screen.

All that should be required from the user is to select criteria from some type of list or list-box to calculate the summary information. This screen must also display the total number of mishaps included in the summary statistics based upon the users selection. Figure 1.2 illustrates an example of the type of output this option should provide:

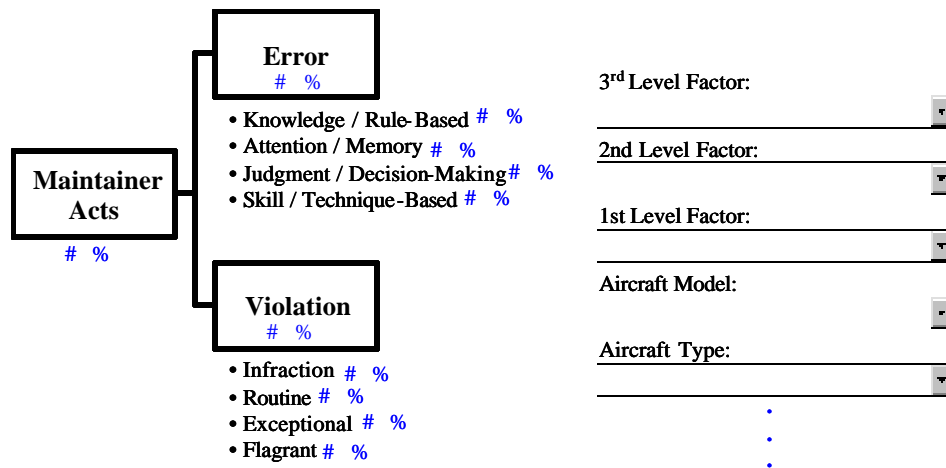


Figure 1.2. Example HFACS Summary.

Details of the Graph option. The graph option should allow users to select various mishap data and dynamically create bar charts for analysis. The user should be able to select any of the following categories to use as X or Y values in the bar chart:

- Aircraft Model
- Aircraft Type
- Organization
- Location
- Mishap Class
- Mishap Type
- Year

Once a category is selected the user will then be able to select a value in that category. For example, if the user selects Aircraft Model, they will be presented with all

the aircraft models within the database to choose from, F-14, F-18, H46 etc. After the initial chart has been viewed, the user should have the option to go back and change selected values or print the chart to a printer.

The Report Option. This option should allow the user print summary reports based upon the following criteria:

- All mishaps
- By aircraft model
- By mishap class
- By mishap type
- By mishap class and type
- By organization
- By location
- All mishaps chronologically
- By 3rd level factor
- By 2nd level factor
- By 1st level factor

Reports should be categorized so as to print in the format of the HFACS-ME taxonomy. Each report should display the total number of mishaps associated with the users selection, the number of mishap factors for each HFACS-ME factor, and the percentage of factor occurrences vs. total mishaps. Figure 1.3 illustrates an example of the type of output this option should provide:

<b>HFACS Summary Report</b>			
<b>Mishaps By Carrier</b>			
<b>As Of: Monday, May 28, 2001 3:14:49 PM</b>			
<b>1 - Air Florida Airlines</b>			
<b>Category</b>	<b>Number</b>	<b>% of Total</b>	
Unsafe Supervisory Conditions (USC)	1	100%	
1	100%	Organization	
0	0%	Hazardous Unsafe Operations	
0	0%	Inadequate Documentation	
1	100%	Inadequate Design	
1	100%	Inadequate Processes	
0	0%	Inadequate Resources	
1	100%	Supervisory	
1	100%	Inadequate Supervision	

Figure 1.3. Example HFACS Report Output.

Details of the Add/Edit Mishap option. This option should provide users the ability to edit any mishap in the database as well as to add new mishaps and factors. Access to the add edit feature must be controlled via a password mechanism. A wizard should be implemented to ensure consistency of each new mishap and factor -- all mandatory data must be provided by the user and validated by program logic for each new mishap and factor. The following data is mandatory for adding a new mishap:

- Aircraft Model
- Aircraft Type
- Service/Organization
- Location
- Mishap Class
- Mishap Type
- Mishap Date
- Description of Mishap

The following data is mandatory for adding a new factor:

- 3rd Level Code
- Factor Description

The user must be able to enter several factors per mishap. The user interface should make use of drop-down boxes to make input as simple as possible.

#### **D. METHODOLOGY**

The methodology used in this thesis research consisted of four phases: 1. Requirements analysis, 2. System Foundation Development/Implementation, 3. HFACS-ME Development/Implementation, and IV) Test and Analysis.

##### **1. Phase I - Requirements Analysis**

This phase consisted of initial analysis of the requirements for both systems. "Use cases" were developed to model domain processes and foster a better understanding of the system foundation requirements. A conceptual model was created to decompose the problem domain in terms of identification of the concepts, attributes, and general associations in the domain. Opportunities for code reuse, common database interface, common schema, and improved performance were investigated. A comparison of

*Microsoft Access* compatible database engines in terms of performance, upgradeability, and scalability was conducted. Finally, an investigation of *Microsoft* development efforts in the areas of *Microsoft Office*, *Microsoft Visual Basic* (VB), and *Microsoft Visual Basic for Applications* (VBA) was conducted to determine best practices for ensuring future compatibility.

## **2. Phase II - System Foundation Development/Implementation**

In this phase, the development effort focused on client/server foundation analysis and implementation. We developed sequence & collaboration diagrams for the typical course of events of each use case related to the client/server foundation. These diagrams were used to illustrate allocations of responsibilities to objects in the system demonstrating how they interact via messages. Next we created Class diagrams based upon these objects (how they connect) and the methods that each software class defined. The end result of this phase was a functioning client / server architecture environment upon which the HFACS-ME program implementation was developed.

## **3. Phase III - HFACS-ME Development/Implementation**

In this phase we utilized the same methods as Phase II to develop working prototypes for the HFACS-ME military and civilian programs. The end product included installation software, an HTML help system, and system documentation.

## **4. Phase IV - Test and Analysis**

This final phase was a wrap up of the research effort. During it we tested our implementation on several different platforms, corrected several minor program deficiencies, and investigated opportunities for future program enhancement.

## **E. ASSUMPTIONS**

Throughout this thesis, We assumed that the reader is familiar with object oriented programming techniques, has a general understanding of the HFACS-ME model, and is familiar with basic Navy and DoD technical terminology.

## **F. DEFINITIONS**

For the purpose of this thesis, the terms Human Factors Analysis Classification System (HFACS) and Human Factors Analysis Classification System - Maintenance Extension (HFACS-ME) will be used synonymously. The ME suffix more accurately

describes the "up to date" implementation of the model which encompasses maintenance related factors. In practice, however, the system is still referred to as HFACS.

All copyrighted material mentioned is © of their respective owners. This thesis does not make any attempt to recommend any of the commercial products mentioned or used in the development of HFACS-ME.

## **G. ORGANIZATION**

This thesis is divided into five chapters. Chapter I presented the problem, background, stated the area of research, and described the methodology, and associated research questions. Chapter II identifies Requirements Analysis through the use of use cases and development of a conceptual model. Chapter III details the development of the client - server foundation of the program. Chapter IV provides similar details for the development of the actual HFACS-ME program. Chapter V provides a summary of research efforts, prototype testing results, conclusions, and recommendations for future enhancements.

THIS PAGE INTENTIONALLY LEFT BLANK

## II. REQUIREMENTS ANALYSIS

### A. OVERVIEW

In this chapter we describe the process used to define functional capabilities, performance & design constraints, system interfaces, and phase allocation of work to the HFACS system. This analysis provided a representation of information and function that was eventually translated into data, architectural, and procedural design. Throughout this requirements analysis process we focused on discovery, refinement, modeling, and specification of the "big picture" HFACS system. We relied heavily on models created using the Unified Modeling Language (UML) and use cases/use case diagrams for gathering operational behavior and determining data content.

The UML is the successor to the various object oriented development tools developed during the 1980's and early 1990's primarily combining the methods of three key pioneers, Booch, Rumbaugh, and Jacobson [Ref. 3]. The UML is referred to as a modeling language rather than a "method" language as it is primarily concerned with using graphical methods over process language to express system design. Much of our analysis in this chapter is graphical in nature and requires knowledge of the UML to appreciate fully.

In addition to using the UML to identify the system features, we investigated several other pertinent areas of the design using more traditional means. Types of data access technologies, compatible programming languages, opportunities for code reuse, and ways to improve performance to name a few. To this end, a comparison of *Microsoft Access* compatible database engines in terms of performance, upgradeability, and scalability was conducted. We also investigated current *Microsoft* development efforts to determine best practices for ensuring future HFACS compatibility. In the end, these steps allowed us to create the overarching conceptual model for our system, allocating work to the remaining design phases as appropriate.

### B. USE CASE ANALYSIS

In order to better understand requirements, domain processes for the HFACS were expressed using use cases and use case diagrams. A use case represents a typical

interaction between a user and the computer system. Use cases are used to capture some user visible function as each one is manifested as some discrete goal for the user. The use cases presented here were created using the most basic of investigation tools such as observation and discussion with people familiar with the current HFACS system. We were not concerned with intricate details of the system when we created these use cases, merely a basic overview of each component/function. Our goal was to learn about how the user really intended to use the system. Descriptions of the various Use Cases are as follows.

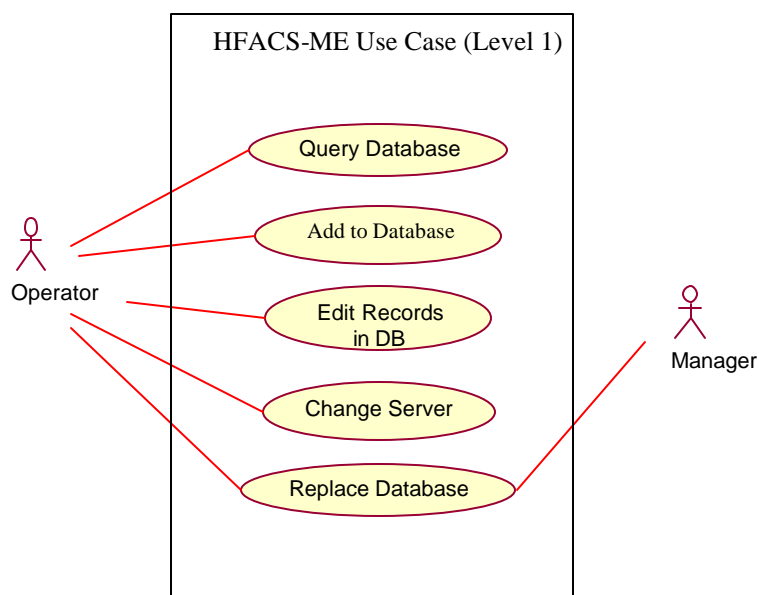


Figure 2.1. HFACS-ME Use Cases (1<sup>st</sup> Level).



## 1. Query Database

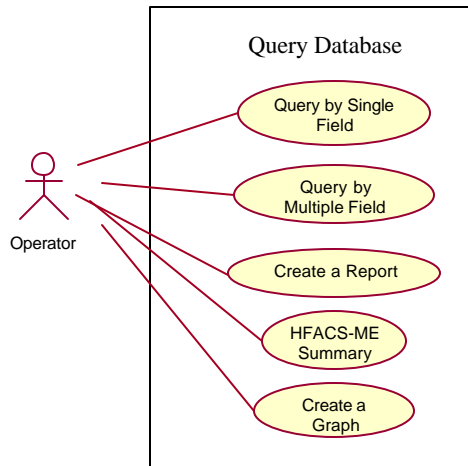


Figure 2.2. Query Database Use Case.

<b>Use Case:</b>	Query Database
<b>Actors:</b>	Operator
<b>Purpose:</b>	To query the HFACS-ME database for information, graphs, and reports
<b>Overview:</b>	The operator needs to be able to query the database for specific information. The operator has the ability to query on a single or multiple fields, obtain summary information, create graphs, and create reports. The operator can perform these functions after the SQL server is started.
<b>Type:</b>	Primary and essential

### a. Query by Single Field

<b>Use Case:</b>	Query by Single Field
<b>Actors:</b>	Operator
<b>Purpose:</b>	To query the HFACS-ME database on single field
<b>Overview:</b>	The operator has the ability to query database on any field of the database that pertains to aircraft mishaps. These queries are pre-built. The HFACS-ME system will retrieve any data item that meets the query conditions.
<b>Type:</b>	Primary and Essential
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the operator selecting to query the database.	2. Presents the operator with general areas to focus the query. For example, aircraft type, aircraft model, location of the mishaps, etc.
3. Operator selects one of the general areas to focus the query	4. Present the operator with a choices to specifically focus the query. For example, all mishaps that involved F14s.
5. Operator selects the specific field to perform the query operation	6. Forms the query and executes the query through the SQL server.
	7. Displays the results to the operator

***b. Query by Multiple Fields***

<b>Use Case:</b>	Query by Multiple Fields
<b>Actors:</b>	Operator
<b>Purpose:</b>	To query the HFACS-ME database on multiple fields
<b>Overview:</b>	The operator has the ability to query database on multiple fields of the database that pertains to aircraft mishaps. These queries are pre-built. The HFACS-ME system will retrieve any data item that meets the query conditions.
<b>Type:</b>	Primary and Essential
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the operator selecting to query the database.	2. Presents the operator with general areas to focus the query. For example, aircraft type, aircraft model, location of the mishaps, etc.
3. Operator selects one or more of the general areas to focus the query	4. Present the operator with choices to specifically focus the query for each general area. For example, all mishaps that involved F14s at Pensacola, FL..
5. Operator selects the specific field of each general area chosen to perform the query operation	6. Forms the query and executes the query through the SQL server.
	7. Displays the results to the operator

***c. Create a Report***

<b>Use Case:</b>	Create a Report
<b>Actors:</b>	Operator
<b>Purpose:</b>	To present the report of aircraft mishaps based on the criteria selected by the operator
<b>Overview:</b>	The operator has the ability to search the database to create reports on aircraft mishaps. These reports will be created based on the specification chosen by the operator. The HFACS-ME system will display the report based on these specifications.
<b>Type:</b>	Primary
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the operator selecting to generate a report.	2. Presents the operator with choices for the type of report to be created.
3. Operator selects one of the report formats (all mishaps, sort by aircraft type, sort by organization, sort by location, or sort in chronological order)	4. Query the database based upon the operator selection to tabulate a report. Display the result to the user.

**d. HFACS-ME Summary**

<b>Use Case:</b>	HFACS-ME Summary
<b>Actors:</b>	Operator
<b>Purpose:</b>	To display the contributing factors to mishaps and the amount the factors in each level contribute to the mishap.
<b>Overview:</b>	The operator has the ability to search the database to create summary information of contributing factors on aircraft mishaps. These summary data will be created based on the specification chosen by the operator. The HFACS-ME system will display the information based on these specifications. All possible factors will be displayed with the percentage of that factor being involved in the accidents.
<b>Type:</b>	Primary and Essential
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the operator selecting to create a summary report of aircraft mishaps.	2. Presents the operator with summary data considering all possible areas (aircraft type, aircraft model, mishap class, etc.).
3. If the operator desires summary data on certain types of mishaps, the operator can select the specific types aircraft mishaps to include in the summary data.	4. Query the database to include only those types of mishaps desired by the operator and present a summary report

**e. Create a Graph**

<b>Use Case:</b>	Create a graph
<b>Actors:</b>	Operator
<b>Purpose:</b>	To display the graphical chart of aircraft mishaps based on the criteria selected by the operator
<b>Overview:</b>	The operator has the ability to search the database to create graphical charts on aircraft mishaps. These charts will be created based on the specification chosen by the operator. The HFACS-ME system will display the chart based on these specifications.
<b>Type:</b>	Non-Critical
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the operator selecting to create a graph of aircraft mishaps.	2. Presents the operator with choices of x and y-axis components. These components are general in nature such as aircraft model, aircraft type, year of mishap, etc.
3. Operator selects one of the general components for the x and y-axis of the graph	4. Present the operator with choices of specific data to be included in the graph. These are specific items such as F14, F18, year 1996, etc.
5. Operator selects the specific item(s) to be included in the graph for both x and y-axis.	6. Query the database to obtain data and display the result in a graphic nature.

## 2. Add to Database

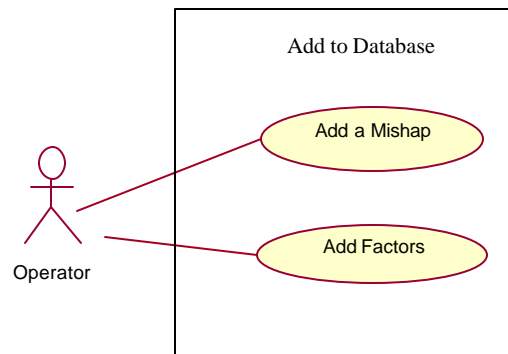


Figure 2.3. Add to Database Use Case.

<b>Use Case:</b>	Add to Database
<b>Actors:</b>	Operator
<b>Purpose:</b>	To add mishap information at the start of the investigation of the accident and to add factors that contributed to the mishap
<b>Overview:</b>	The operator has the ability to input into the database data that pertains to the mishap. The operator also has the ability to input contributing factors that led to the mishap. The operator can perform these functions after the SQL server is started.
<b>Type:</b>	Primary and essential

### a. Add a mishap

<b>Use Case:</b>	Add a Mishap
<b>Actors:</b>	Operator
<b>Purpose:</b>	To add mishap data into the database
<b>Overview:</b>	As new aircraft mishaps occur, the operator has the ability to add mishap data into the database. The data includes date of the mishap, description, cost, type of aircraft, model of the aircraft, location, category of the mishap, and organization involved.
<b>Type:</b>	Primary and Essential
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the occurrence of a new aircraft mishap. Operator selects to add a new mishap into the database.	2. Requests all pertinent information for this mishap. The information required includes: data of mishap, aircraft type, mishap type, mishap class, organization, category, location, whether any crewmen were injured, damage to the aircraft, and description of the incident.
3. Operator provides the information required.	4. Adds the mishap incident into the database and inform the operator of the successful transaction.
	4. If the record could not be added, inform the operator of the failed transaction.

### b. Add Factor

<b>Use Case:</b>	Add Factors
<b>Actors:</b>	Operator
<b>Purpose:</b>	To add factors contributing to the mishap into the database
<b>Overview:</b>	As an investigation commences, factors leading to the mishap may be discovered. As the factors are discovered, the operator has the ability to add contributing factor data into the database for a specific mishap. The data includes the factors from all three levels of categories (first order, second order, and third order).
<b>Type:</b>	Primary and Essential
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the discovery of a contributing factor to the aircraft mishap. Operator selects to add a new mishap into the database.	2. Queries the operator for the factor from the first order factors
3. Operator selects one factor from the first order factors that contributed to the accident.	4. Queries the operator for the factor from the second order factors. These factors depend upon the first order factor selected.
5. Operator selects one factor from the second order factors that contributed to the accident.	6. Queries the operator for the factor from the third order factors. These factors depend upon the second order factor selected.
7. Operator selects one factor from the third order factors that contributed to the accident. Operator also provides a brief description of the factor.	8. Updates the database by inserting the new factor in the database for the record containing this aircraft mishap. Queries the operator for additional factors.
9. Operator indicates he has additional factors or not	10. Repeat sequences 6 to 10 if additional factors need to be added.
<b>Alternative Courses</b>	
	10. Operator indicates that new factor is in a different second order factor category or different first order factor category. Repeat sequences 2-10 as needed.

### 3. Edit Records in Database

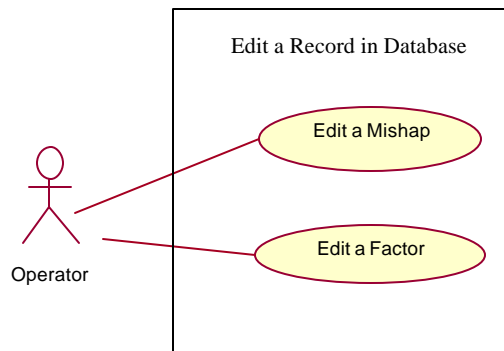


Figure 2.4. Edit a Record in Database Use Case.

<b>Use Case:</b>	Edit
<b>Actors:</b>	Operator
<b>Purpose:</b>	To edit the mishap information as the investigation of the accident gains information and to edit the factors that contributed to the mishap
<b>Overview:</b>	The operator has the ability to change the data in the database data that pertains to the mishap. The operator also has the ability to edit the contributing factors that led to the mishap. The operator can perform these functions as additional information is obtained.
<b>Type:</b>	Primary and essential

*a. Edit Mishap*

<b>Use Case:</b>	Edit a Mishap***
<b>Actors:</b>	Operator
<b>Purpose:</b>	To edit a mishap incident from the database.
<b>Overview:</b>	As new information is discovered or an error in the data is discovered about an aircraft mishap incident that already exists in the database, the operator has the ability to edit the mishap data.
<b>Type:</b>	Primary and Essential
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the discovery of new information or error in existing information of an aircraft mishap incident that exists in the database. Operator requests a query to search for the aircraft mishap incident in question.	2. Requests the incident number or the type of query to search for the incident. For example, select all mishaps that occurred at this location during this year.
3. Operator selects the incident that needs to be edited.	4. Display all pertinent information about this incident. Display the incident number, data of mishap, aircraft type, mishap type, mishap class, organization, category, location, whether any crewmen were injured, damage to the aircraft, and description of the incident, any factors that contributed to the incident that has been entered previously.
5. Operator makes adjustments to the data item that needs to be corrected or created.	6. Update the database with the new information. Inform the operator of success or failure of the update.

***b. Edit Factor***

<b>Use Case:</b>	Edit Factors***
<b>Actors:</b>	Operator
<b>Purpose:</b>	To edit factors in an aircraft mishap incident from the database.
<b>Overview:</b>	As an error in the data is discovered about a contributing factor to an aircraft mishap incident that already exists in the database, the operator has the ability to edit the factor data.
<b>Type:</b>	Primary and Essential
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the discovery of an error in existing information of an aircraft mishap incident that exists in the database.	2. Display the mishap incident and all of its contributing factors.
3. Operator selects the factor or factors that needs to be edited.	4. Display the information about the factor. Display factor description and its first order factor grandparent, second order factor parent.
5. Operator makes adjustments to the description or indicates that the factor's parent needs to be changed.	6. Query the operator for first order factor, second order factor, and third order factor as necessary.
7. Selects the first order factor, second order factor, and third order factor as necessary.	8. Update the database with the new information.

**4. Change Server**

<b>Use Case:</b>	Change Server
<b>Actors:</b>	Operator
<b>Purpose:</b>	To change the SQL server
<b>Overview:</b>	The operator has the ability change SQL server without closing the HFACS-ME program
<b>Type:</b>	Primary
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the operator choosing to change the active server.	2. Disconnect to the current server. Request the address and name of the new server.
3. Type in or select a new server	4. Establish connection to the selected server.
	5. Inform the operator of successful change or failed change.

## 5. Replace the Database

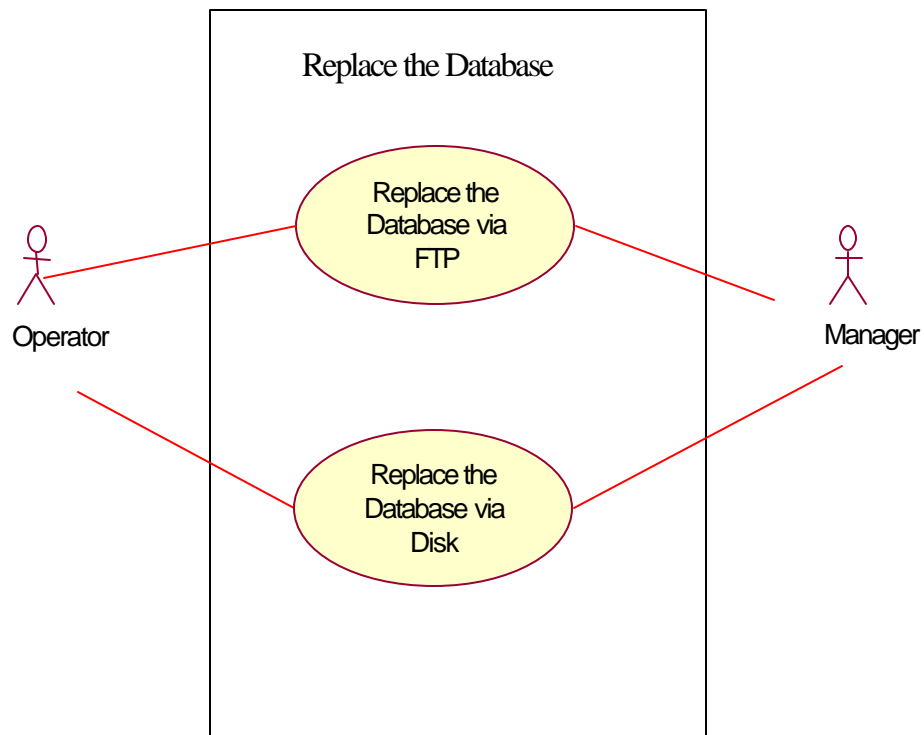


Figure 2.5. Replace the Database Use Case.

<b>Use Case:</b>	Replace Database
<b>Actors:</b>	Operator, Manager
<b>Purpose:</b>	To replace or update the existing database with a new database
<b>Overview:</b>	Once the manager has obtained a new HFACS-ME database, the operator has the ability to update or replace the existing database with the new database. The operator has the ability to perform this via FTP or via disk operation.
<b>Type:</b>	Primary



**a. Replace the Database via FTP**

<b>Use Case:</b>	Replace the Database via FTP
<b>Actors:</b>	Operator, Manager
<b>Purpose:</b>	To replace the existing database with new current version of the database via FTP mechanism.
<b>Overview:</b>	As many mishaps are added to the database, the local databases may not be the same throughout the location. To bring all databases to the same version, a new database can be uploaded through the network.
<b>Type:</b>	Primary
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the new database being available at a central site provided by the manager. The manager informs all clients that a new database is available for upload.	
2. The operator directs the system to upload the new database.	3. Disconnect all concurrent users on the local system.
	4. Creates a backup of the existing database and stores it in the file system.
	5. Downloads the database from the central site and stores it on the local system.
	6. Uploads the database and starts the server.
	7. Inform the operator of the successful or failed operation.

**b. Replace the Database via Disk**

<b>Use Case:</b>	Update the database from National HQ Master Files
<b>Actors:</b>	Operator, Manager
<b>Purpose:</b>	To replace the existing database with the master database
<b>Overview:</b>	The HFACS-ME system's database can be updated (replaced) by the master database to bring all organizations to common database. This can be done through network download or via disc operation.
<b>Type:</b>	Primary and Essential
<b>Typical Course of Action</b>	
<u>Actor Actions</u>	<u>System Response</u>
1. This use case begins with the master database being available provided by the manager. The manager informs all clients that a new database is available for upload or a disc is available with the database.	
2. The operator directs the system to upload the new database.	3. Request the operator for the method of the replacement operation.
4. The operator provides network operation method.	5. Disconnect all concurrent users on the local system.
	6. Creates a backup of the existing database and stores it in the file system.
	7. Downloads the database from the central site and stores it on the local system.
	8. Uploads the database and starts the server.
	9. Inform the operator of the successful or failed operation.
<b>Alternative Courses</b>	
4. The operator provides disc operation method.	4.1 Request file location from the operator.
4.2 Operator provides the database location (directory)	

### **C. CLASS-RESPONSIBILITY-COLLABORATION (CRC) CARDS**

The use cases identified above are really very high-level execution scenarios for the HFACS program. The next step in our analysis required us to take these scenarios and identify the objects in each one. We wanted to focus on the actions that these objects would be responsible for so that we could develop our classes from them. As part our literature review, we came across Ward Cunningham and Kent Beck's [Ref. 4] class-responsibility-collaboration (CRC) card methodology. The CRC card method for developing classes uses 4" X 6" index cards to map responsibilities to objects. A "responsibility" is a description of the purpose of the class. The idea is to try to get away from a description of data and processes by capturing the purpose of the class in a few sentences. The choice of a card was deliberate - we chose not to allow more than what would fit on a single card to represent a single object.

This decision to use CRC cards proved very fruitful. Our CRC cards have the class name in the upper-left hand corner, a bullet- list of responsibilities under it in the left two-thirds of the card, and the list of other classes needed to fulfill that responsibility in the right third of the card. This simple method of assigning responsibilities gave us great insight into the links between classes, but still at a high level -- we did not get bogged down in the details. Most useful was the ability to discuss many different design possibilities without writing a line of code. By accenting responsibilities instead of data and methods we were able to develop a fairly thorough understanding the behavior for each class. By grouping the cards together, we could begin to visualize what would become our packages (actually, dynamic link libraries). We could identify classes that had been given too much responsibility and reassign those responsibilities to other classes, where appropriate. The most interesting discovery made in creating these cards was the greatly apparent distinction between the database platform objects and HFACS program objects. It was very obvious to us that there were certain responsibilities that were specific to each client and others that were the identical for all clients. Our CRC cards can be found at Appendix A.

#### D. MICROSOFT ACCESS & DATABASE ENGINES

The Aviation Safety School system requirements specified a *Microsoft Access 2000* implementation of the new HFACS system. In our opinion, *Microsoft Access 2000* is a very powerful and deceptively complex program that can function as a database engine, database client, or both. This section discusses the different functionalities of *Access 2000* and the reasoning we used to determine its HFACS implementation.

*Microsoft Access* has built-in functionality to create desktop applications with forms, reports, and embedded support for *Visual Basic for Applications* (VBA). In addition, the data in an *Access* database can be manipulated using several different programming languages, active server pages (ASPs) via the web, and via third party add-in tools. A key feature of *Access* over other databases and development tools is its ease of use - it is a very effective rapid application development (RAD) platform. When compared with databases such as *Oracle*, *Access* can be (depending upon implementation options) magnitudes simpler to use for creating similar applications.

A new feature of *Access 2000* that made it appealing for the HFACS project is the ability to use more than one type of database engine. A database engine is the part of a database management system (DBMS) that actually stores and retrieves data. *Access 2000* provides support for both the *Microsoft JET* database engine and the *Microsoft SQL Server* engine. This is a key distinction. *Access* formerly allowed only one choice of database engine: *JET*. The main problem with *JET* is that it is not a client/server capable engine. It is primarily a file server. This means that anytime a client wants to request something from a *JET* database everything has to be done on the client-side. The result is a lot of network traffic and unacceptable response times for more than only a handful of simultaneous users. With the release of *Office 2000*, however, *Microsoft* provided a royalty free version of the *SQL Server* engine capable of running on a desktop computer. This change allowed an *Access* solution the ability to operate as a stand-alone application using the same engine as the full version of *SQL Server* -- it is 100 percent compatible because it is the same engine. Upgrading from a desktop application to a server-based application is no longer an issue because the engine is the same.

One confusing aspect of the standalone engine is the difference in naming conventions between various versions of *SQL Server*. The *SQL Server 6.5 and 7.0* compliant version is called the *Microsoft Data Engine* (MSDE), while the *SQL Server 2000* compatible variant is called *Microsoft SQL Server Desktop Edition*. Both versions of the engine should offer the same functionality when used with *Access*, but this is not entirely true, as will be described in Chapter III. For the remainder of this thesis, in order to provide greater emphasis on the distinction between full *SQL Server* and the Desktop editions, we will refer to both the *SQL 7.0* and *2000* versions of the desktop engine as *MSDE* unless otherwise stated.

In our research we found that it is a very common requirement to have a *JET* based database and desire to migrate it to a more robust database engine - namely *Microsoft SQL Server 7.0* or *SQL Server 2000*. We also found that automated migration tools designed to port *JET* databases over to *SQL server* are useful only for very simple databases. We experimented with using the *Microsoft Access* "Upsizing wizard" on both the military and civilian versions of the existing HFACS system with very poor results. Structured Query Language written in *Access* using *JET* did not transfer correctly. Functions written in VBA did not transfer correctly. In addition, the data types used by *JET* are different from those in *SQL Server* and they did not transfer properly. Finally, *Access* uses "queries" in place of stored procedures and queries did not transfer at all. To put it simply, the *JET* database engine is not scaleable and was ruled out as a viable option for the new HFACS very early in the requirements analysis process.

## **E. DATA ACCESS TECHNOLOGIES**

Following the decision to use *Microsoft SQL Server* as the database engine for HFACS, we realized that the majority of our personal experience with database design dealt with *Microsoft JET*. Our review of *MSDE* indicated that it had a lot to offer in terms of use with *Access* and *Visual Basic*. For example, the desktop engine supports record-level locking, transaction logs, operating-system integrated security under Windows 2000, and many other advanced features of full *SQL Server* (like replication) -- all from *Visual Basic* and VBA. In fact, we found that the *SQL Server* engine actually had a plethora of options, most formidable of which was the selection of programming

interface to access the data in it. We feel that most of this complexity is unnecessary and directly related to *Microsoft's* proprietary implementation of object-oriented data access methods.

In the early 90's, the Object Management Group defined methods for the Common Object Request Broker Architecture (CORBA) that were designed to create an industry standard for universal data access using object-oriented methods [Ref. 7]. *Microsoft*, however, has its own competing standards called the Distributed Component Object Model (DCOM) and Component Object Model (COM). The COM is a binary standard which defines how an object should present itself to the system, regardless of programming language used. COM programs are referred to as "components." Generally COM components are compiled into Dynamic Link Library (.DLL) format. The DCOM is an extension of COM, which allows object creation to span over a network in a client-server environment, hence the "distributed" prefix. The *Microsoft SQL Server* engine supports DCOM, COM, and other legacy data access technologies. The most prolific of which are: Object Linking & Embedding for Databases (OLE DB), ActiveX Data Objects (ADO), Open Database Connectivity (ODBC), Data Access Objects (DAO), Remote Data Objects (RDO), SQL Direct Management Objects (SQLDMO), and several lesser variants. Each of these technologies offers various functionalities. Selection of the method(s) that HFACS would use was a critical decision as the wrong choice could impose limitations in functionality and/or compatibility later in our development process [Refs. 7, 14]. A complete study of all these object models is beyond the scope of this paper, so only a brief description of the major ones is provided here. For more information consult the *Microsoft Universal Data Access* web-site at: <http://www.microsoft.com/data/>.

## **1. OLE DB**

Object Linking & Embedding for Databases comprises a model consisting of data providers and data consumers. The providers contain and expose data, while the consumers use data and services. Basically, OLE DB is capable of providing data from a variety of sources by using *Microsoft* COM. OLE DB is just a set of these COM components designed specifically to access data as producers and consumers. This is

particularly powerful because developers can build their own components and include them as part of the interface -- as long as they use development tools compatible with *Microsoft COM*. OLE DB provides the underlying layer of abstraction that enables most of the other technologies in the *Microsoft Universal Data Access* initiative. Through this layer of separation, OLE DB enabled applications can improve data access by allowing dynamic binding to lots of different data stores. A very interesting capability associated with this technology, is that once bound, OLE DB components can provide services, like SQL querying, against data sources that normally cannot perform the processing themselves (like flat text files). Figure 2.6 illustrates the architecture.

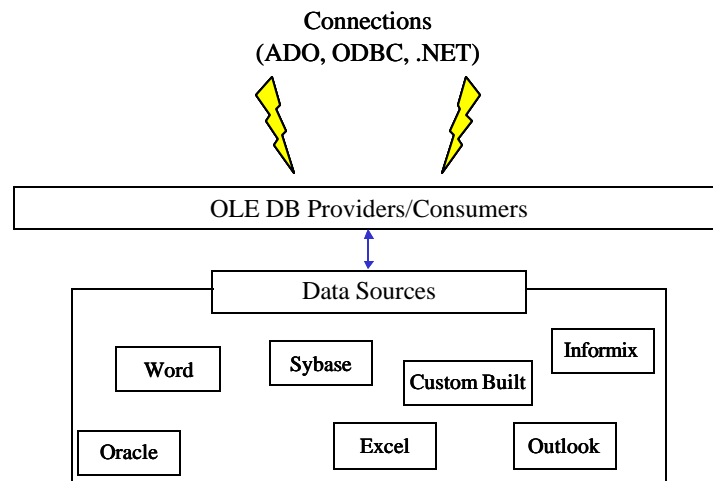


Figure 2.6. OLE DB Architecture.

## 2. ADO

ActiveX Data Objects support a variety of needs, including the creation of front-end database clients and middle-tier applications that provide the "business rules" for interaction with a back-end databases or other applications such as an Internet browser. *Microsoft* touts the ADO programming model as "the best of the existing *Microsoft* data access programming models [Ref. 15]." This is primarily due to its relative ease of use, speed, low memory overhead, small disk footprint, and tight coupling with OLE DB. Connection objects in ADO are easy to use as are command objects and recordset objects. Where OLE DB is concerned more with accessing data sources, ADO is

concerned with mapping the data to visual controls, like data grids and combo-boxes -- which compliments *Microsoft* visual development languages such as *Visual Basic* and *Visual C++*.

### **3. ODBC**

The Open Database Connectivity (ODBC) interface has been around for many years. ODBC uses SQL to access data based upon drivers. Drivers are vendor specific interfaces between an application and a specific brand of database. ODBC drivers exist for everything from ASCII text files, mid-size databases like *FoxPro*, up to enterprise databases like *Oracle*. A problem with ODBC is that not all drivers implement all the functions of ANSI SQL, so the level of support you get can vary based upon vendor.

### **4. DAO**

Data Access Object technology was developed in 1994 to allow *Visual Basic 3.0* to access and manipulate data in local or remote databases. DAO was the first object-oriented interface that exposed the underpinnings of *Microsoft JET* and allowed Visual Basic developers to directly connect to Access tables - as well as other databases - through ODBC. This was a very powerful feature, but *Microsoft* is currently only providing support for it so that applications can be backwards compatible. When working with *JET*, there are certain functions that DAO can provide that the other technologies cannot, but the risk of obsolescence is great and this technology should be avoided. DAO is suited best for either single-system or small multi-user applications.

### **5. RDO**

*Microsoft* first released the Remote Data Objects model in 1995 to support *Visual Basic 4.0* [Ref. 14]. RDO was developed to provide object-oriented methods to access high-end ODBC relational data sources like *SQL Server* or *Oracle*. RDO provides the properties and methods needed to access "more complex" stored procedures and result sets. The idea behind RDO was it could save *Visual Basic* programmers a great deal of time by allowing them to access the RDO interface without directly coding the ODBC API. In the past, RDO has proven to be a popular interface for the large relational databases. Similar to DAO, however, *Microsoft* is currently only providing support for it so that applications can be backwards compatible.

## 6. SQLDMO

The SQL Distributed Management Object interface is a proprietary feature of *SQL Server*. The SQLDMO.DLL communicates with SQLSVC.DLL (the database abstraction module), which accesses ODBC32.DLL, which in turn implements the *SQL Server* ODBC driver. As evidence of its power, if you are familiar with the *SQL Server Enterprise Manager*, much of the functionality you see in it was implemented with SQLDMO [Ref. 21]. SQLDMO provides management functions for *SQL Server* at a very low level. For example, instances of the server can be started and stopped, regardless of connection state -- you can stop the server even if users are logged on. You can add users, set permissions, add databases, and tables. In addition to management functions, SQLDMO can be used to run stored procedures and perform data access type functions. The problem with SQLDMO is that it is not easily accessible via the Internet and therefore is undesirable for other than management functions. Microsoft is phasing out SQLDMO in favor of Windows Management Instrumentation (WMI) type interfaces. The purpose of WMI is to define a non-proprietary set of enterprise management specifications. These specifications allow management information to be shared between applications that run on different operating systems. Luckily, WMI currently prescribes standards that are backwards compatible with SQLDMO.

The brief overview you just read is just the tip of a very large iceberg when it comes to evaluating *Microsoft* data access technologies. We found this part our research very troublesome and overly complicated. In the end, we discovered that *SQL Server* comes with its own native OLE DB provider, which means that *SQL Server* does not have to be paired with a web-server to provide support for multiple tier database solutions - as long as you choose OLE DB compliant technologies to access the data. Multiple tier solutions will be discussed more in chapter IV, but we mention it here to demonstrate that scalability concerns were addressed in all aspects of our design. Since OLE DB is natively part of the SQL environment and such a big part of *Microsoft's* current *Universal Data Access* strategy, it made sense to use it over ODBC. In addition, since RDO and DAO both seemed to be legacy technologies whose functionality is slowly being consumed by ADO, it made sense to use ADO wherever possible. We recognized,



however, that in dealing with MSDE as the engine for desktop versions of HFACS, we would need access to management functions beyond the capability of ADO. For these needs we would use SQLDMO to access the SQL engine through its ODBC driver.

These facts, coupled with the *Microsoft* and several third party recommendations to use ADO and OLE DB led us to select them as our primary data access methods wherever possible. The new HFACS system actually uses four of these technologies: OLE DB, ADO, ODBC, and SQLDMO, which will be expanded upon throughout this paper.

#### **F. PROGRAMMING MICROSOFT ACCESS AND SQL SERVER**

Microsoft *Access* has a history of notorious incompatibilities between versions. Since 1993, *Access* has undergone fundamental changes with each new release. *Access 2.0* applications used *Access Basic* rather than VBA and did not convert to *Access 95* format. *Access 95* implemented many new technologies and did not always convert to *Access 97* format. Our personal experience with trying to upsize old versions of HFACS from *Access 97* to *Access 2000* clearly demonstrated that there were problems with it as well. Based on this history alone we concluded that the next version of *Access* would no doubt have similar problems. The requirement for an *Access* based solution from our sponsor was firm and at this point seemed somewhat constraining. The search for a method to lessen the impact of version changes became paramount.

As mentioned earlier, *Access* has embedded support for *Visual Basic for Applications*. The SQL engine, however, is accessible via any language capable of creating COM objects. This realization presented a unique option for mitigating the effects of future *Access* version incompatibilities. Using *Visual Basic* or C++, we could design *ActiveX* object-oriented components that encapsulated much of the code that would normally be written within *Access*. These compiled components would reside outside of *Access* theoretically making them less susceptible to version changes and maximizing potential for code reuse. *Access* would just be a client shell and all business logic would be placed in these external components. The beauty of this approach is that the RAD methods of *Access* used to create forms, reports, and controls were still available. In addition, this approach is in keeping with the migration path of a small-

scale application to a larger enterprise level one using OLE DB and DCOM. The location of the external components (either client-side or server-side) would define the architecture of the system (3-tier or n-tier).

Removing the business logic from *Access* allows HFACS to grow by enabling modification of component code without making changes (or many changes) to the client code in *Access* or the database elements in *SQL Server*. Since code in the components is compiled, changes in versions of the programming language used to create them are much less significant in over the lifespan of the program. We knew this would be especially significant for HFACS because of *Microsoft's* upcoming release of new technologies like *C#* and *Visual Basic .NET*. Regardless of the technology changes associated with these upcoming releases, current versions of *C++* and *Visual Basic* should still be able to create compiled components compatible with new versions of *Access* and *SQL Server*. Since code is removed from the front and back-end Microsoft products, we believe that components are much less likely to suffer from versioning issues. The disadvantage of all this of course, is the inherent complexity in creating these components. As alluded to in the previous sections, the vast array of features in *Access* and *SQL Server* make creating components to take advantage of these products a very ambitious goal.

Based on our decision to implement components, our next decision involved selection of a COM compatible programming language. In keeping with the requirement for a Microsoft based solution our choices were either *Visual Basic 6.0* (VB) or *Visual C++*. Both *C++* and VB are capable of implementing the four data access technologies that we knew we would need. Since *Access* provides inherent support for VBA and *Visual Basic 6.0* is a superset of this technology, VB could provide a single language for use in both *Access* and the components. *C++*, on the other hand, offered greater support in terms of threading (which will be discussed further in Chapter III). A major disadvantage of *C++*, however, was its added complexity in a program designed for RAD. In the end, the idea of a using VB in all coding for HFACS was truly the key factor in weighing advantages and disadvantages. Our final choice for programming language was *Microsoft Visual Basic 6.0* using Service Pack 5.

## G. MICROSOFT DEVELOPMENT EFFORTS

Despite our vision of immunity from version changes in *Visual Basic*, *Access*, and *SQL Server*, we conducted a review of *Microsoft* development efforts to ensure our design would comply with the product manufacturers existing interoperability guidance. Our findings:

### 1. Access 2002 [Ref. 16]

According to *Microsoft*, *Access 2002* databases (based on the *JET* database engine) will work with two database file formats — *Access 2000* and *Access 2002*. In *Access 2002*, you will be able to modify data and make design changes to an *Access 2000* database. During an *Access 2002* rollout, *Microsoft* recommends using the *Access 2000* file format. In this mixed environment, both *Access 2000* and 2002 users will have a default file format of 2000. Features that are new in *Access 2002* will be available when using an *Access 2000* file in *Access 2002*, but will not be available when the same file is used in *Access 2000*. When a file is opened in *Access 2000*, any functionality specific to *Access 2002* is simply ignored. In a mixed file format environment, *Microsoft* strongly recommends design and update of all databases using *Access 2000*. If designed with *Access 2002* using the *Access 2002* file format, users cannot open the database with *Access 2000*. Although not specifically stated by *Microsoft*, it is assumed that features compatible with a *SQL Server* engine will be similarly compatible with both 2000 and 2002 file formats, therefore HFACS should be compatible.

In addition to the file format changes, *Access 2002* will support both ANSI-89 SQL (also called JET SQL) and ANSI-92 SQL, which have new and different features. The two ANSI SQL query modes, ANSI-89 and ANSI-92, are not compatible. Since HFACS uses *SQL Server* as its database engine, our implementation already uses ANSI-92 SQL and this should not be a factor. Finally, Office 2002 will come with the *SQL Server 2000 Desktop Engine*, not MSDE 1.0. Although both of these database engines will be able to coexist on a single computer, they are not 100% compatible. This will be discussed in more detail in Chapter III).

## 2. Visual Basic.NET [Ref. 17]

*Microsoft Visual Basic.NET* will be a complete rebuild of the current version of VB. *Visual Basic.NET* will take a major step toward making *Visual Basic* a fully featured object oriented language with new features including full object-oriented design capabilities and free-threading. Several limitations of VB 6.0 that VB .NET is planned to remedy were problems in our development of HFACS. Workarounds will be discussed in subsequent chapters. For this reason, we believe upgrade of the HFACS components to VB .NET when it is released will be desirable. *Microsoft* states that "*Visual Basic.NET* will open and upgrade *Visual Basic 6.0* projects to *Visual Basic.NET* technologies, but in most cases you will need to make some modifications to your projects after bringing them into *Visual Basic.NET*." [Ref. 18] *Microsoft* recommends a host of considerations to enable future upgrade to VB .NET [Ref. 18], the most significant of which are discussed below:

- Use of early binding of variables. Objects should be declared as the data type that they really are rather than as type Object. In VB .NET late-bound objects can introduce problems when resolving default properties. Additionally, the Variant data type is replaced by Object, so Microsoft recommends discontinuing its use. Our HFACS code uses early binding wherever possible.
- Use of ADO for data access. VB .NET will provide support for DAO, RDO, and ADO in code with some modification. However, Visual Basic.NET does not support DAO and RDO data binding to controls. Since HFACS does not use RDO or DAO, modifications should be relatively simple.
- Avoidance of the Double data type for storing dates. HFACS uses the Date data type for dates.
- Avoidance of fixed-length strings in user-defined types. HFACS does not implement any user defined types, only user defined Classes.
- Resolve Parameterless Default Properties using dot-notation. HFACS uses complete object property references, so this should not be a problem.
- Use of enumerated constants instead of underlying values. Wherever possible HFACS uses the enumerated constants, however, there are some instances where zero resolves to null for which zero has no enumerated value.
- Use special syntax for declaring fixed arrays. The current method for declaring fixed arrays (e.g. myArray(5) As Integer) will not work with

VB.NET. Syntax in the following form should be used instead: Dim MyVariable As MyType; ReDim MyVariable.MyArray(5) As Integer. HFACS uses this recommended syntax.

- Avoid Legacy Features. Because they have been removed from the language, the following keywords should be avoided: **Def<type>**, **Computed GoTo/GoSub**, **GoSub/Return**, **Option Base 0|1**, **VarPtr**, **ObjPtr**, **StrPtr**, and **Lset**. HFACS uses none of these keywords except **On Error GoTo** for error handling – for which there is no *Microsoft* recommendation to remedy.

As previously stated, the *Visual Basic 6.0* format should remain viable as long as versions of *Access* and *SQL Server* provide support for COM components -- so migration of HFACS to VB .NET isn't mandatory, just desirable at some point. Interesting to note that a parallel situation exists for the *Visual C++* programming language, as *Microsoft* has similar plans for migration to C# which also implements .NET technology. For this reason, our selection of *Visual Basic* as programming language remained intact.

### 3. SQL Server

*Microsoft* released the *SQL Server 2000* family of products less than six months prior to our development effort. No service packs existed and there was no publicly accessible information related to follow on versions available at that time.

## H. THE CONCEPTUAL MODEL

The use cases and CRC cards developed in our requirements analysis effort coupled with our research of data access technologies, programming languages, and trends in *Microsoft* products enabled us to develop a vision of our HFACS system. Armed with this information we set about creating a conceptual framework for the design of the system. As part of this process we inferred the following:

- HFACS should consist of a *Microsoft Access* client application using external compiled components to encapsulate business processes wherever possible. This would provide greater opportunity for code reuse and mitigate the effects of version changes in *Access*.
- HFACS would implement the *SQL* database engine and therefore should be developed so as to connect to an instance of *MSDE* as well as true *SQL* server. In order to facilitate differences in these connections, a component would be needed to perform management functions such as installation of the programs, installation of the database, logon options, and starting and stopping the server. Management functions of this depth have to be performed using *SQLDMO* and are specific to each client, therefore, this

component must also reside on the client. Figure 2.7 illustrates the conceptual model for this component.

- The business processes associated with the actual manipulation of the objects in the database were not specific to each client. Based upon our review of DCOM and COM, we recognized that to provide scalability for HFACS, further investigation into which technology to implement would be needed. What we could conclude, however, is that these processes needed to be encapsulated in a component separate from the connection component. Furthermore, this component should not include any user forms or GUI components making it more abstract and versatile. Figure 2.8 illustrates the conceptual model for this business-logic component.

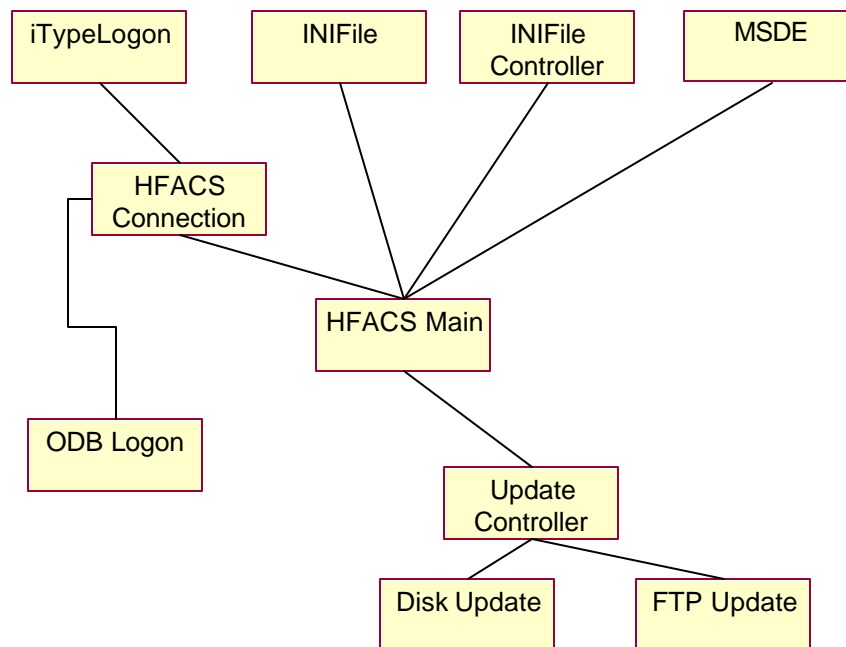


Figure 2.7. Conceptual Model for the Connection Component.

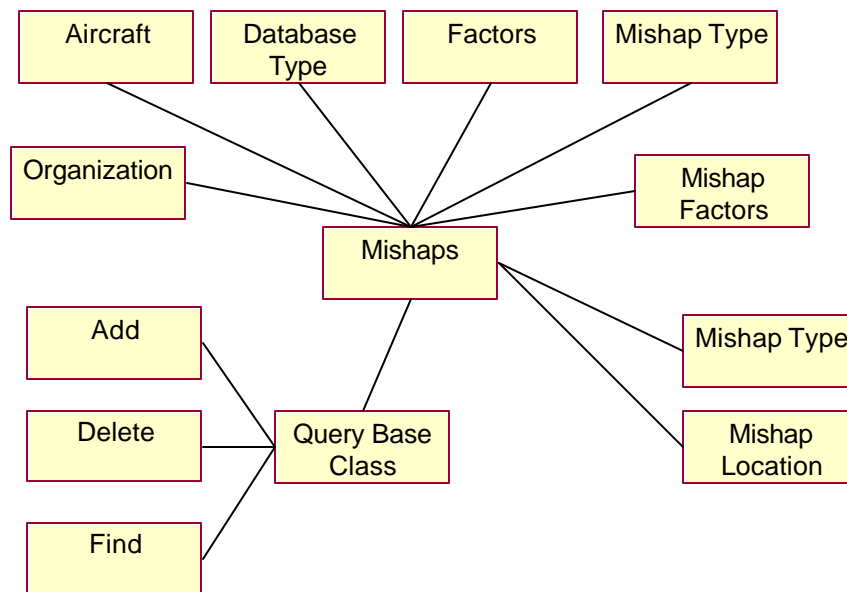


Figure 2.8. Conceptual Model for the Business -Logic Component.

From these findings it became clear that the development effort should be divided into two phases. Phase I should focus on development and implementation of the HFACS Connection component. Phase II should do the same for the HFACS business logic component. The development of the connection component was to be executed first because it would involve creating the foundation and environment for the business logic component to operate in. In addition to creation of the connection component and the inherent connection functions, Phase I would involve creating the installation programs needed to deploy and configure all the pieces of this operational environment on a wide array of platforms supporting various editions of *SQL Server* and *Windows* operating system. If possible, this component should be capable of working with different versions of *SQL Server* as well as different editions.

Upon completion of Phase I, we envisioned a much broader understanding of the *SQL* engine, which would help us in developing database schema and selecting an architecture (DCOM, COM, 3-tier, or n-tier) for the business logic component in phase II. The high-level conceptual architecture is illustrated below.

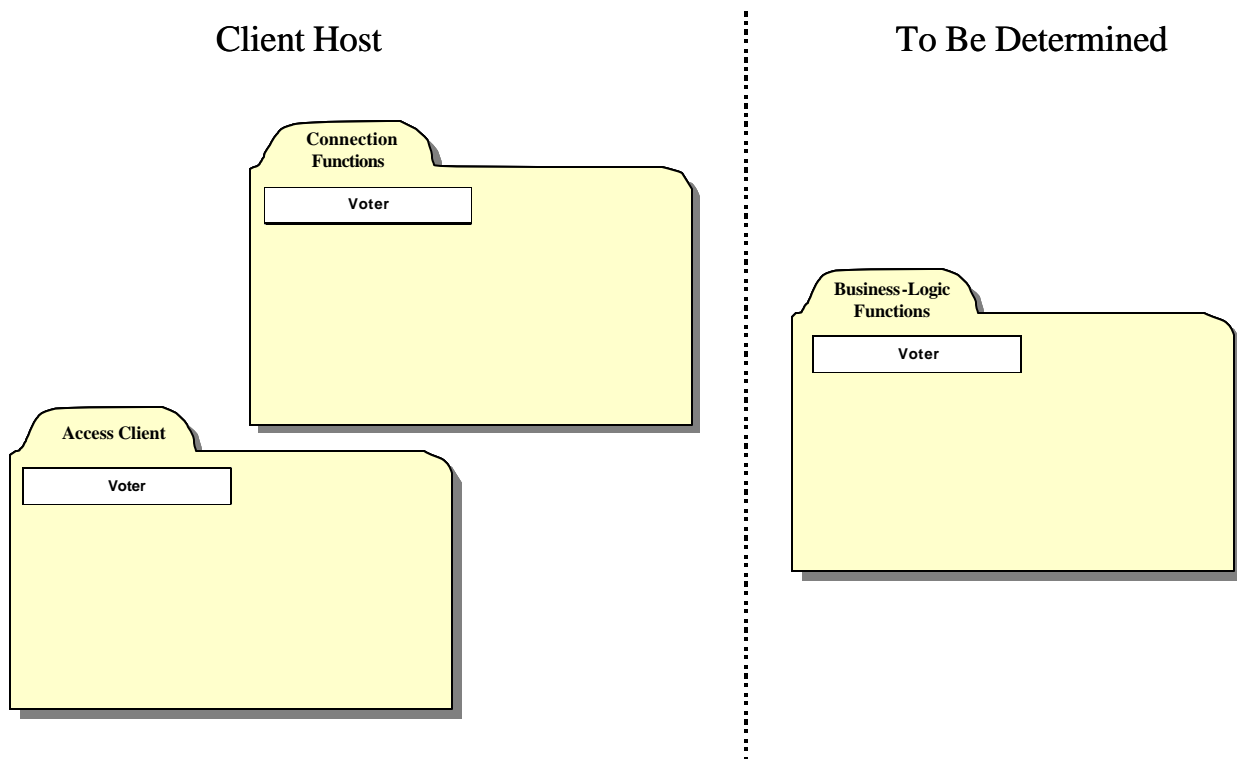


Figure 2.9. Conceptual Architecture at the End of Requirements Analysis.

Within phases, we planned to use Spiral Development Model (SDM) [Ref. 19] techniques. The SDM made the most sense to us because although requirements had been fairly well defined for HFACS, there was still a substantial amount of risk associated with our lack of experience with *SQL Server*, object oriented programming with *Visual Basic*, and the *Component Object Model*. In addition, we knew that in the course of our development process, requirements might change. For instance, new requirements for the commercial aircraft version of HFACs might arise. In addition, there was a good chance that one of the other development groups could make requirement changes. The SDM provides built-in methods for mitigating these risks through its use of development stages. Each stage would be a normal development project producing a superset of the prior stage and yet a subset of the final system. Planning for each successive stage would be structured to exploit the experiences of the former stages and to reduce perceived risk factors in the current and future iterations.



### **III. HFACS CONNECTIVITY COMPONENT DEVELOPMENT**

#### **A. OVERVIEW**

This chapter provides a detailed description of the design and implementation of the HFACS connectivity component. The component was constructed as an ActiveX dynamic link library, which is included as a reference in the *Access* client program. *Access* Client programs that are used in conjunction with the *SQL Server* engine are called Access Data Projects and can be identified by their ".adp" file extension.

We began development of this component by refining its conceptual model through interaction diagrams using the UML. Two types of interaction diagram were used in this process: sequence diagrams and collaboration diagrams. Both types of diagrams allowed us to refine our conceptual model into class diagrams. Once class diagrams were in place, we identified stages of spiral development and began coding.

The culmination of this phase was the HFACS installation program and connection dynamic link library incorporating the functionality needed to install *Microsoft Access* Runtime, the *SQL Server* engine, the *Access* client data project file, an initialization file used for maintaining client installation settings, a separate compiled FTP server, and the methods to install and replace instances of the HFACS database.

#### **B. SEQUENCE DIAGRAMS**

Our first step in refining the conceptual model was to create Sequence diagrams for the typical course of events of critical use cases in order to better understand system behavior. The sequence diagrams that follow illustrate the actor interactions and the operations initiated by them, as well as, their order.

## 1. Change Server

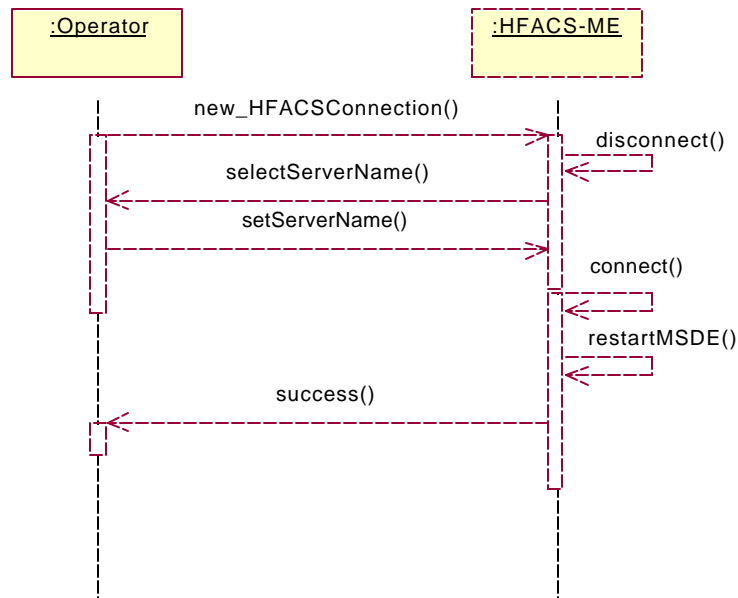


Figure 3.1. Change Server Sequence Diagram.

## 2. Replace the Database via FTP

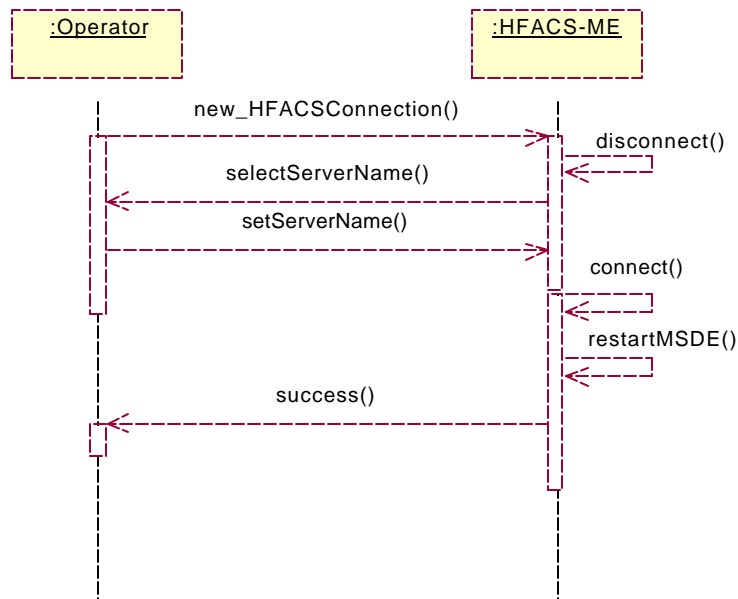


Figure 3.2. Replace the Database via FTP Sequence Diagram.

### 3. Replace the Database via Disk

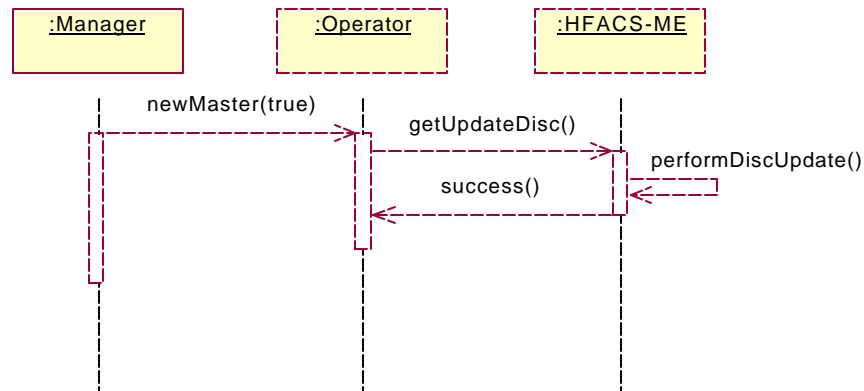


Figure 3.3. Replace the Database via Disk Sequence Diagram.

## C. COLLABORATION DIAGRAMS

Analysis of our Sequence diagrams allowed us to create Collaboration diagrams to illustrate allocation of responsibilities to objects in the system, specifically demonstrating how they interact via messages. The diagrams that follow provided the level of detail needed isolate the key messaging functions between objects in the component.

### 1. Change Server

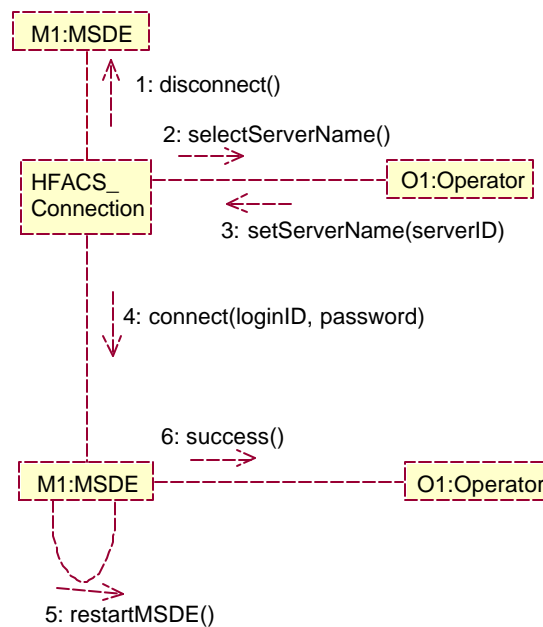


Figure 3.4. Change Server Collaboration Diagram.

## 2. Replace the Database via FTP

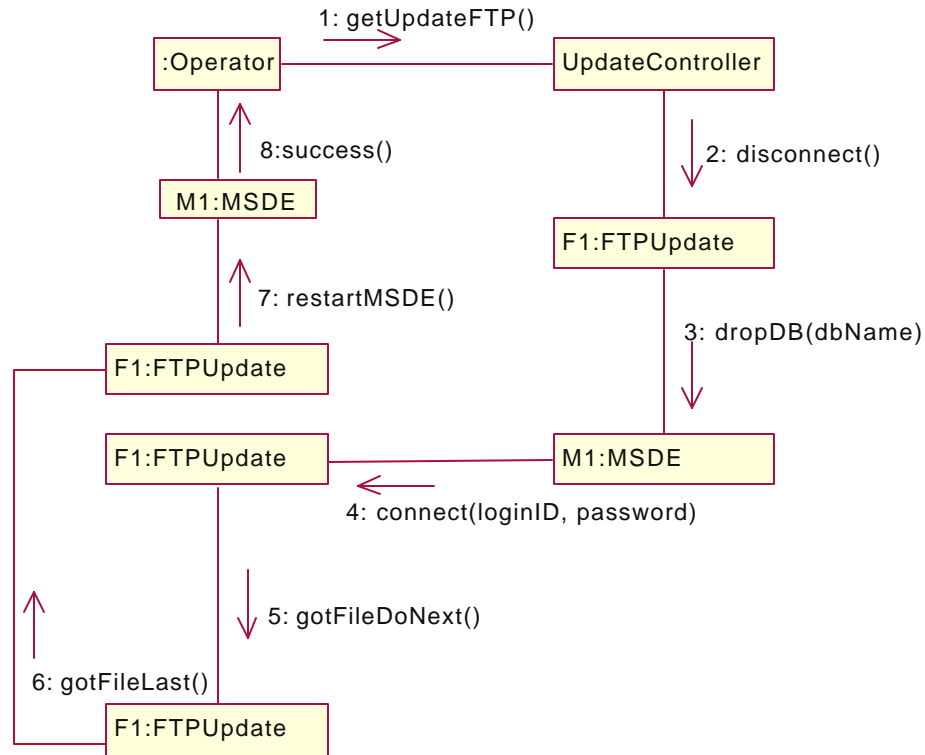


Figure 3.5. Replace the Database via FTP Collaboration Diagram.

## 3. Replace the Database via Disk

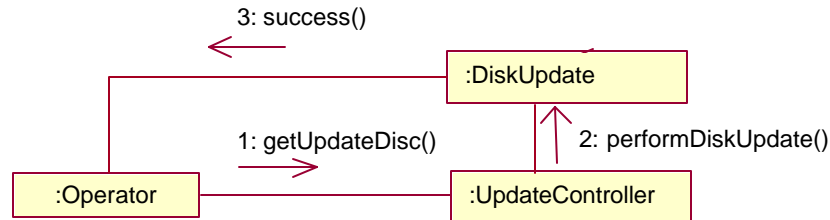


Figure 3.6. Replace the Database via FTP Collaboration Diagram.

## D. CLASS DIAGRAMS

The information gleaned from the Collaboration diagrams, empowered us with the knowledge needed to refine our conceptual model. Figure 3.7 illustrates an intermediate level view of the key classes. The descriptions that follow provide abridged definitions and explanations for these key classes. They are provided here to document

their general functionality in prose format and provide a basis for subsequent discussion of development issues. Detailed HFACS connection component class diagrams illustrating all methods, as well as, complete descriptions of the actual classes the can be found at Appendix B & C, respectively.

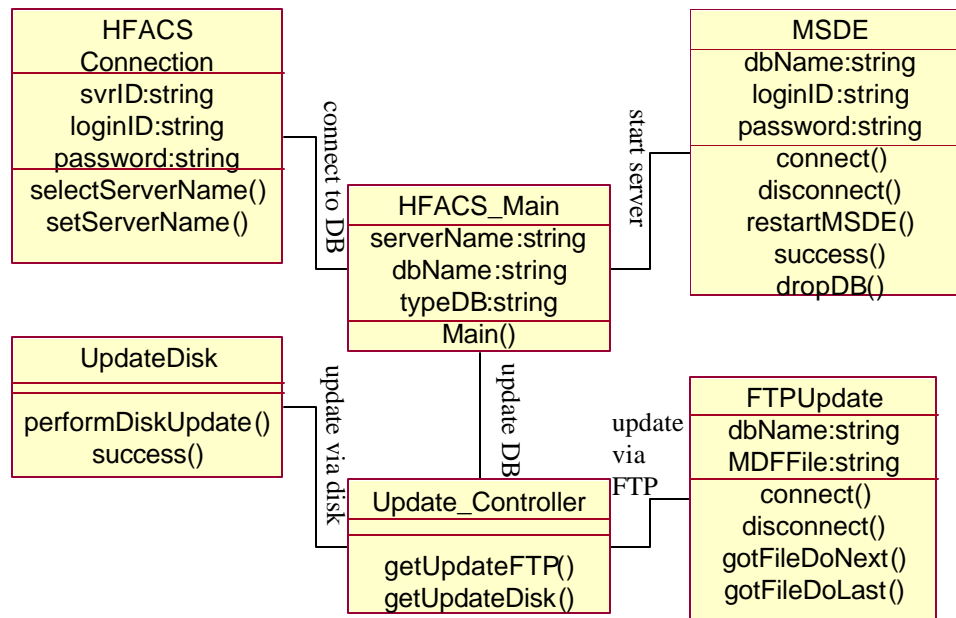


Figure 3.7. Interim Class Diagram.

### 1. HFACS Connection Class

The HFACS Connection class encapsulates the functionality of the entire component and provides the interface for all other classes. It is the only class with public members accessible from outside of the component. Instantiating the HFACS Connection class allows the calling program to connect to a SQL server by passing connection arguments. Connection arguments can be input via logon dialog box or by reading stored values from an initialization file (HFACS.ini). The connection process logic is capable of starting the SQL server using SQLDMO objects encapsulated by the MSDE class. Instances of this class also provide public methods for the calling program to change the server in mid operation of the HFACS-ME system and for replacing the HFACS database with updated versions via disk/FTP.

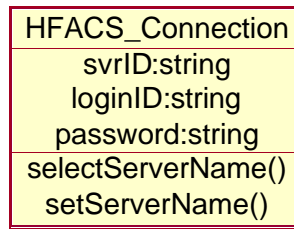


Figure 3.8. Class Diagram for HFACS Connection.

## 2. HFACS\_Main Class

This class is the “Main” class for the component. *Visual Basic 6.0* requires a Main class for all dynamic link library components function. For those familiar with C++, it is similar to “Program Main ” – required for runtime execution. It is instantiated any time the .dll is called (when the program starts running). In the context of our use, it is also used to store global variables such as the SQL server name, database name, and type of database.

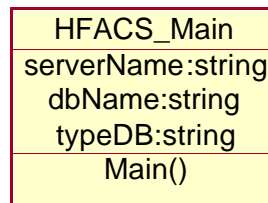


Figure 3.9. HFACS\_Main Class Diagram.

## 3. UpdateController Class

The UpdateController class is the business logic class responsible for controlling the FTPUpdate class and the UpdateDisk class. It facilitates the manipulation of forms and other objects when replacing the database via FTP or the disk method.

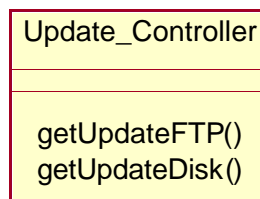


Figure 3.10. Class Diagram for UpdateController Class.

#### 4. UpdateDisk Class

UpdateDisk is responsible for performing an update of the HFACS database from a disk/network share.

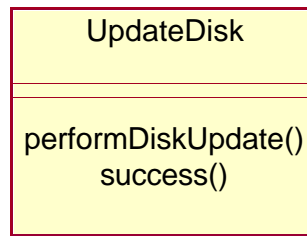


Figure 3.11. Class Diagram for UpdateDisk Class.

#### 5. FTPUpdate Class

This class is responsible for performing an update of the HFACS database via FTP. Since a SQL Server database is comprised of two files (HFACS.mdf & HFACS\_log.ldf), it has methods that allow it to monitor download and installation of each file, separately.

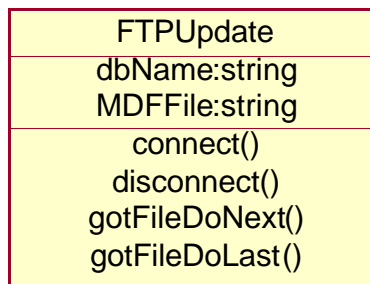


Figure 3.12. Class Diagram for FTPUpdate Class.

#### 6. MSDE Class

The MSDE class performs all SQLDMO object manipulation. It is responsible for starting the *MSDE* or *SQL Server* engine, ensuring that the HFACS database is installed, and managing database updates. Additionally, it provides the functionality to attach and detach the database files fed to it by the UpdateDisk and FTPUpdate classes.

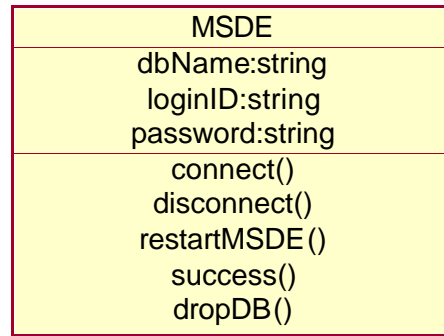


Figure 3.13. Class Diagram for MSDE Class.

## E. IDENTIFICATION OF SDM STAGES

As discussed in the previous chapter, we utilized the Spiral Development Method (SDM) as a guide to control risk throughout the component development process. Before beginning coding of the HFACS DLL component we had to choose which version of *SQL Server* to develop our application with. At the time of this writing, *Microsoft SQL Server 2000* had only been commercially available for approximately six months. *Microsoft SQL Server Version 7.0* was definitely the more mature database engine with plenty of available documentation and support on the Internet. Both versions offered support for running as a desktop engine dedicated to a single instance of HFACS, as a single server supporting large numbers of clients, or as part of a cluster of servers supporting entire enterprises. Both versions also offered support for multiple processors, discretionary security, transactions, and triggers. Based on our *Microsoft's* prior tendency to make new programs backward compatible, we chose *SQL 7.0* and MSDE 1.0 as our development version of the engine. We felt that migration of the code to include *SQL 2000* functionality might be difficult, so we planned to do it as part of a separate stage. Based on this decision we identified the following three stages of cyclical development:

- Stage 1 - Creation of an HFACS Connection component compatible with the SQL Server 7.0 engine.
- Stage 2 - Modification of the component to make it compatible with both the SQL Server 7.0 and 2000 engines.
- Stage 3 - Creation of installation programs to install and configure the component and all related files on Windows 95 or newer platforms.



## F. IMPLEMENTATION - STAGE 1

Since the HFACS Connection component is a stand-alone compiled ActiveX dynamic link library, it was developed using the *Microsoft Visual Basic 6.0* Integrated Development Environment (IDE) program. *Microsoft Access* has its own native development IDE and cannot use the true VB 6.0 IDE. We knew that switching back and forth between the two IDEs and trying to find faults would be difficult. Our strategy for avoiding this was to create the classes of the component and a separate "test" program to validate class behavior all within the VB IDE environment. Visual Basic provides support for this in the form of a "Visual Basic Group" (.vbg) project. This plan worked well. Using the .vbg we could place two separate projects in the same workspace allowing them to run in the IDE at the same time. By initially testing class behavior within the VB IDE instead of from the external *Access* environment, we were able to isolate problems to their sources much more quickly -- without all the IDE switching. In addition, when the time came to test the component with *Access*, since we knew it worked in VB, we immediately knew the problem was either on the *Access* side, or in the interface.

The first challenge we faced in our coding was the inability of *Visual Basic 6.0* to provide true inheritance. We were aware that *Visual basic* only provided "has a" or compositional inheritance, but our initial coding efforts proved this limitation difficult to adapt to. Luckily, *Visual Basic* does provide support for secondary interfaces to classes using the *Implements* keyword. We attempted to use base classes and interfaces wherever possible to make up for the lack of true inheritance. This provided many benefits, most notable was the ability to fix a bug in a base class and have all the derived classes "inherit" the change through the interface -- without having to edit code in the other classes. We only had to modify code in derived classes when we changed the interface of the base class, added new properties and methods, or deleted existing ones.

The first truly unforeseen difficulty we came across was the inability of Visual Basic to define a constructor with parameters. In more mature object oriented programming languages, a constructor can be defined in the class module and executed whenever a new instance is created. Because you define the syntax of the constructor

method, you can force the client code to pass arguments that are needed to create the object, or return an error if the required information is not provided. In fact, several constructors can be defined which take different parameters. In VB, there are no constructors. Instead, there is a class *initialize* event which can be programmed to ensure all objects start in consistent state. The problem is that the class *initialize* event cannot be overridden and it cannot take arguments. This is a serious shortcoming in VB that will be corrected in *VB.NET*. To work around this, we used pseudo-constructor methods wherever possible. To create a pseudo-constructor, a public function was defined in a globally accessible module (the HFACS\_Main class). These functions were given names like "New\_MSDE " with function prototypes including optional parameter lists. Optional parameter lists have to be used because functions cannot be overridden. When these functions are called, they perform two operations: 1) creation of an instance of the class and 2) execution of a Friend "init" function from the class which takes matching optional parameters. If this sounds confusing, it is. Let me give a specific example. Consider the following code excerpt from the MSDE class:

```
*****
                          Psuedo Constructor for the MSDE Class
*****
```

This public function is placed in a globally accessible module. Notice that it takes 10 optional arguments.

```
Public Function New_MSDE(Optional sUser As String, _
    Optional sPassword As String, _
    Optional sSvrName As String, _
    Optional sMDFName As String, _
    Optional sDBName As String, _
    Optional sInstDirectory As String, _
    Optional sAutomaticLogon As String, _
    Optional sFirstRunCheck As String, _
    Optional sNTAuth As String, _
    Optional sTypeDB As String)
```

The first operation performed is the creation of an instance of the MSDE class. This invokes the class\_Initialize event of the MSDE class, which can take no arguments as parameters. In order for this to work, the oMSDE object variable must be declared prior to calling the function. In this case, it was declared as a reusable package (DLL) level variable.

```
Set oMSDE = New MSDE
```

Next, the optional arguments are verified. If they are missing, then predefined values stored in a .DLL level instance global variable are used. This ensures all instances of the object are created in consistent state.

```
If IsMissing(sUser) Then sUser = gStrUID
If IsMissing(sPassword) Then sPassword = ""
If IsMissing(sSvrName) Then sSvrName = gStrServerName
If IsMissing(sMDFName) Then sMDFName = gStrDatabaseFileName
If IsMissing(sDBName) Then sDBName = gStrDatabaseName
If IsMissing(sInstDirectory) Then sInstDirectory = gStrAppPath
If IsMissing(sAutomaticLogon) Then sAutomaticLogon = gStrAutoLogon
If IsMissing(sFirstRunCheck) Then sFirstRunCheck = gStrFirstRun
If IsMissing(sNTAuth) Then sNTAuth = gStrNTauth
If IsMissing(sTypeDB) Then sTypeDB = gStrTypeDB
```

Next, since member functions can have parameters, the Friend function member of the MSDE class instance just created is called and the parameters are passed to it.

```
oMSDE.Init sUser, _
    sPassword, _
    sSvrName, _
    sMDFName, _
    sDBName, _
    sInstDirectory, _
    sAutomaticLogon, _
    sFirstRunCheck, _
    sNTAuth, _
    sTypeDB
```

```
End Function
```

Now lets look at the pertinent functions in the MSDE class.

```
*****  
MSDE Class Code Extract  
*****
```

This code defines the Class\_Initialize event, which is really a default no-argument constructor. It basically populates the module level variables.

```
Private Sub Class_Initialize()  
  
    sUser = gStrUID  
    sPassword = gStrPWD  
    sSvrName = gStrServerName  
    sMDFName = gStrDatabaseFileName  
    sDBName = gStrDatabaseName  
    sInstDirectory = gStrAppPath  
    sAutomaticLogon = gStrAutoLogon  
    sFirstRunCheck = gStrFirstRun  
    sNTAuth = gStrNTauth  
    sTypeDB = gStrTypeDB  
  
End Sub
```

Here we see the custom "Init" function called by the psuedo-constructor that results in the mimicked behavior of a constructor that takes arguments.

```
Friend Sub Init(sPassedInUser As String, _  
    sPassedInPassword As String, _  
    sPassedInSvrName As String, _  
    sPassedInMDFName As String, _  
    sPassedInDBName As String, _  
    sPassedInInstDirectory As String, _  
    sPassedInAutomaticLogon As String, _  
    sPassedInFirstRunCheck As String, _  
    sPassedInFirstRunAfterUpdate As String, _  
    sPassedInTypeDB As String)  
  
    sUser = sPassedInUser  
    sPassword = sPassedInPassword  
    sSvrName = sPassedInSvrName  
    sMDFName = sPassedInMDFName
```

```
sDBName = sPassedInDBName  
sInstDirectory = sPassedInInstDirectory  
sAutomaticLogon = sPassedInAutomaticLogon  
sFirstRunCheck = sPassedInFirstRunCheck  
sNTAuth = sPassedInFirstRunAfterUpdate  
sTypeDB = sPassedInTypeDB
```

End Sub

This psuedo-constructor mechanism worked well in the pure visual basic environment, however, when we compiled the DLL and tried to create an HFACSConnection object using the psuedo-constructors in the global modules of the package from *Microsoft Access* -- it didn't work. As it turns out, global modules of a compiled DLL only have package level scope. So, they are not visible from *Access* because *Access* is external to the package. This was not a showstopper, but it reduced the effectiveness of the psuedo-constructor method and caused heavier reliance upon global variables.

The next unforeseen problem we encountered in our implementation was VB's lack of support for free threading. In free threading, each thread can access the entire process's data area and all threads share the applications global variables. In the future, *Visual Basic .NET* will provide free threading. The main problem with free threading is that you have to keep track of all the shared resources, including variables. You can very easily end up with a deadlock situation. *Visual Basic 6.0* tries to provide an easier method for dealing with multiple threads through the use of "apartment" threading. Apartment threading, however, only provides different threads for instances of entire components. For example, three different users could access the HFACS component from different computers and each would receive their own instance of the objects in the DLL. These instances of the DLL would each have their own thread and reside in their own "apartment." Each apartment has its own set of variables and code from one apartment can't access that of another apartment. This effectively eliminates the scheduling problems associated with shared global variables that are very problematic in more other programming languages. The problem with this approach is that you can't

directly launch a new thread from within your apartment. Here are the specifics of our problem.

As part of the HFACS connection component's functionality, it needed to be able to connect to an external FTP server and download replacement copies of the HFACS database files. The FTP class we used to provide this capability wraps the functionality of the WININET.DLL file that is part of all Windows platforms. The WININET.DLL provides API hooks to the operating system for Internet connectivity. This solution worked well with one exception. When the user downloaded a file, the HFACS program became blocked waiting for the `getFile` method of the `cFTP` class to successfully download the database update. As a result, no screen updates could occur within HFACS. If the user launched an instance of another program while the FTP was downloading, and then minimized the application to view the status of the download, the HFACS screen would not redraw. The user was left with a screen full of white unpainted controls -- it was impossible to determine if the FTP was still in progress or if the computer had locked-up and become unresponsive. To work around this problem required a rather complex implementation implementing a "callback" technique.

The callback mechanism works like this: the client application calls a method from an external component compiled as an executable that will take a relatively long time to execute, it passes a reference to an object defined in the client application and the external component stores this reference in local variable. This variable is then used to call back to the client to inform it that something has occurred. Since the external component is an ActiveX executable file, it runs in its own process space. To make use of this functionality, the `cFTP` class was removed from the HFACS Connection component and an interface class was designed for it. These two classes were then compiled as a separate executable FTP server. A reference to the compiled file was included in the HFACS Connection component and a callback class was created using the "Implements" keyword. Figure 3.14 illustrates a high-level overview of the concept.

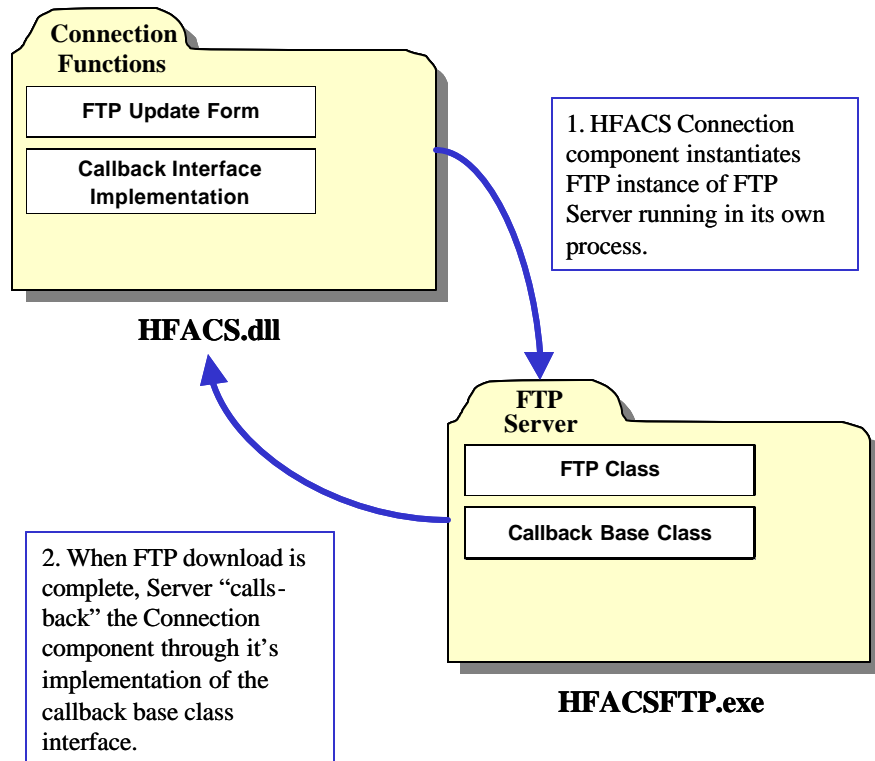


Figure 3.14. OLE DB Architecture.

The callback function worked well for us. Since the FTP file download was now running in its own process space, the screen in the HFACS component was free to redraw. This implementation also resulted in better performance of the cancel button on the FTP form, which became much more responsive to user interaction.

## G. IMPLEMENTATION - STAGE 2

After several weeks of enhancements and code revisions, the component appeared to be working well enough for us to begin contemplating modifications to enable it to work with the *SQL Server 2000* engine. This marked the beginning of stage 2. As briefly mentioned in chapter II, our literature review indicated that both versions are supposed to be forward compatible -- that is, a SQL 7.0 database file can be read by SQL 2000 [Ref. 20] and that the existing SQLDMO model is compatible with a *SQL Server 2000* database, less the new features of *SQL Server 2000* [Ref. 21].

Our first step was to install *SQL Server 2000 Standard Edition* on our development computer and test our existing code with it. *SQL Server 2000* installed a

new version of SQLDMO, made some changes to file locations, and offered several new options for dealing with new support for server "instances", but otherwise the *SQL Server 2000* installation was very similar to that of *SQL Server 7.0*. Using our Visual Basic test program we successfully used our existing HFACS Connection component to start the server, detach a database file, and stop the server. The first problem we encountered was with attaching a database file. As it turned out, the new version of SQLDMO required use of brackets ("[" and "]") to separate arguments in its *attachDB* method. The old version of SQLDMO would accept either spaces or brackets. This was a simple fix.

The next problem proved more difficult to solve. The *doConnect* method of our HFACSConnection class provides the functionality to create and test a connection to a new server. The *doConnect* method created an instance of the frmODBLLogon class, which in turn used SQLDMO to verify the connection information specified by the user in the logon dialog box. This was accomplished by: 1) attempting to start and connect to the server; and 2) looping through database objects on the server to confirm existence of the HFACS database. For some reason, the *SQL 2000* version of SQLDMO will not allow starting of a remote server. This proved troublesome, as the ability not just to connect to a remote instance of the database, but also to start it, was a desirable feature. An exhaustive search of the Internet and newsgroups failed to yield any valuable information related to this problem. As best as we could discern, this problem is related to the added features associated with the ability of *SQL Server 2000* to create multiple separate "instances" of SQL servers on the same machine. These instances listen for clients on different ports. In *SQL Server 7.0*, port 1433 was used for all network traffic, unless specifically changed to another port by a database administrator. Apparently, the new version of SQLDMO doesn't know which port to use and does not use the default of 1433. As a workaround, we modified the MSDE and frmODBLLogon classes in our component to use ADO instead of SQLDMO for verifying remote connections. Although this workaround does provide the functionality to validate a user's logon information, the capability to actually start a remote SQL server was lost.



Now that we had modified our component to work with both versions of the SQL engine from a pure *Visual Basic* environment, we were ready to compile and test it using *Access* as a front-end. In our premier test, we encountered a host of errors, including:

- All existing stored procedure names were displayed with ";1" at the end.
- None of the stored procedures could be run without receiving an error stating that the stored procedure could not be found.
- If you tried to use the security management functions of *Access* to add logons and users to SQL Server, an error message stating "components failed to load or initialize" was displayed.
- You could not create or design tables, database diagrams, or stored procedures without errors.

Since our component worked perfectly in the pure Visual Basic environment, we quickly concluded that there were significant compatibility problems between *Access 2000* and the new *SQL Server 2000* engine. Luckily, an Internet query identified that *Microsoft* had already acknowledged these problems and addressed them through two fixes. The first fix was *Office 2000 Service Release 1/1a*. This service release seemed to fix all the problems except for the ability to create or design tables, database diagrams, or stored procedures without errors [Ref. 22]. These remaining incompatibilities were fixed by the second patch called the *Access 2000 and SQL Server 2000 Readiness Update* [Ref. 23].

In the end, we were able to modify the component and install these two patches to get everything working in the SQL 2000 environment. The requirement for installation of the two patches is extremely unfortunate as it complicates system requirement validation for users. Aggravating matters, the *SR-1/1A* update for Office cannot be bundled with the HFACS distribution due to Copyright. In addition, the *Access/SQL Update* does not have a user friendly installation program. It requires users to manually unpack and copy files to program directories, making it clumsy and much less professional in terms of ease of use. Nonetheless, the patches do provide the functionality that *Microsoft* proclaims and the HFACS Connection component will work in a *SQL 2K* environment if they are both properly installed.

## H. IMPLEMENTATION - STAGE 3

At the end of stage 2, the HFACS Connection component had been tested on several platforms with both *SQL Server* engines as back-end data sources. We were confident that it was ready to be bundled into an installable set of programs capable of deployment on any computer running a Windows 95 or newer *Microsoft* operating system. This bundle of programs would need to install our component, the FTP server, the initialization file, either the SQL 7.0 or 2000 engine, the *Access* project file, and all the associated library reference files. In addition, we desired to bundle *Microsoft Access Runtime* as part of the package. *Microsoft Access Runtime* is a stripped-down version of *Access 2000* that allows developers to distribute *Access* based solutions to users without the requirement for *Access* to already be installed on the user's machine. *Access Runtime* is only available as part of the *Microsoft Office Developer* version of *Office 2000*. We realized that three setup programs would be needed: one for our component and its associated files, one for the SQL 7.0 engine, and one for the SQL 2000 engine.

We began by creating a setup program for our HFACS component and related files. The first step was to identify libraries and files that needed to be included with our compiled code in order for it to run. Since we had tracked program dependencies as part of development, this was relatively easy. Next, we needed to determine where on the users machine to install the files in order for HFACS to find them at runtime. The following matrix identifies the files and locations, less those associated with installation of either of the two SQL engines:

Filename	Function	Client Directory Location
comdlg32.ocx	Provides objects needed to use the file open/close dialog	Application path
gif89.dll	Provides objects for displaying animated .gif files	Application path
HEACS.adp	The HEACS Access client project	Application path
HEACS.bmp	Splash screen	Application path
HEACS.dll	The HEACS Connection component	Application path
HEACS.ico	HEACS program icon	Application path
HEACS.ini	HEACS initialization file	Application path
HEACS.mdf	Initial distribution of the HEACS database	Application path
HEACS.mdf.old	Back-up copy of the database	Application path
HEACS_log.ldf	Initial distribution of the database log file	Application path
HEACS_log.ldf.old	Back-up copy of the log file	Application path
HEACSETP.exe	The FTP server	Application path
MSCOMCT2.ocx	Provides objects for common controls like the status meter	Application path
MSCOMCTL.ocx	Provides objects for common controls like buttons and text boxes.	Application path

Figure 3.15. File Install Locations.

The *Visual Basic Enterprise Edition* includes a *Package & Deployment Wizard* for creating Setup programs for compiled applications. *Microsoft Office Developer* includes a similar *Package & Deployment Wizard* for creating Setup programs for custom Access solutions. Both versions of the Wizard allow specification of file install locations in much the same format as the table of Figure 3.15. The *Developer* version has the added ability to bundle Access runtime -- which is the only way runtime can be distributed. This is effectively a measure to prevent non-owners of the *Developer* edition of *Office* from copying the runtime files and including them in their distributions. For this reason, we used the *Developer* edition of the *Package & Deployment Wizard* for creating our *Visual Basic* based setup programs.

The source code for the Setup program used by the *Package & Deployment Wizard* is included with both *VB Enterprise* and *Developer*. Creation of our setup package would require modification of the *Developer* version code in two ways. First, a capability needed to be added to write changes to the *hfacs.ini* initialization file as part of the program install. This was needed so that the HFACS Connection component could determine the application path of the *Access* project without reliance on the *Access* project to pass this information. This was a trivial matter, as all we were required to do was add the *INIFile* class to the Setup program project with some simple code to write the application path to a key as part of setup. The second modification was also trivial. In order to include a custom icon as part of the distribution, a few lines of code had to be added to the Setup program source code as defined in Lynn Shanklin and Brady Deal's article, "*Distributing Custom Icons with Your Microsoft Office 2000 Applications*" [Ref. 24].

Our modified setup program compiled perfectly using the *Office Developer Package & Deployment Wizard*. Installing it on several machines, however, we found several inconsistencies associated with the different Windows operating systems. All the previous problems identified with the SQL 2000 engine were present, as was a new bug dealing with "Multiple System Files Out of Date" (see Microsoft Knowledge Base article # Q279764). The good news was that every deficiency we discovered was corrected in *Office Service Release 1/1a* and the *Access 2000 and SQL Server 2000 Readiness Update*

were installed on the machine in the proper sequence. These findings highlighted the importance of stressing application of these patches to *Microsoft* products in our final user documentation.

The next step dealt with creating setup files for both versions of the SQL engine. The *SQL Server 2000 Desktop Engine* setup program provided by *Microsoft* was designed for distribution as part of a bundle. Microsoft has conveniently packaged the required files in a directory with a customizable initialization file for setting application specific options. The program installs with a single screen, displaying a graphical status bar that indicates progress. All we had to do was add parameters to the initialization file forcing the engine to install itself using mixed mode security settings. This ensures that when the engine is installed on a computer running a *Windows 2000* operating system, the HFACS Connection component will still be able to access the engine using the default "sa" logon with a blank password. Of course the user can change these settings using the osql.exe command line management tool or by upgrading to a "full" version of Microsoft SQL server after installation.

Creating an installation setup program for the SQL 7.0 desktop engine was much more complex. The setup program provided by Microsoft for the SQL Server 7.0 engine is a full-fledged GUI program with multiple screens requiring the user to enter detailed information about SQL security settings, ODBC data sources, and other information that really requires a background in SQL server administration to understand. We hypothesize that a user installing the desktop engine will generally be someone interested running a standalone version of the HFACS-ME program, so we wanted MSDE to install automatically. The solution to this problem was to create an unattended installation file, which recorded all the "answers" to the wizard dialogs. In this manner, the setup program could be launched from an MSDOS batch file specifying command line options to run with the settings specified in the unattended installation file. This, however, presented another problem. The DOS window remained open for the approximately eight minutes it took to install MSDE without providing any feedback to the user -- just a black screen and lots of disk activity. We wanted to create a GUI status bar screen similar to one used by the *SQL Server 2000 engine*. In keeping with the desire to keep all

our code in Visual Basic, we pondered several possible implementation scenarios, all of which were deemed undesirable due to the lack of free threading in VB. Since the program would be very small, we opted for a *Visual C++* implementation.

## **I. SUMMARY**

In the end, we were able to create code that would support both versions of the SQL engine, but maintenance of two separate installation programs was undesirable. In addition, the *SQL 7.0* and *SQL 2000* engines utilize different versions of the SQLDMO model and the related files are located in different default directories. This meant that we would have to develop two different versions of our *Access* client to accommodate the different reference file locations. Alternatively, we could programmatically manipulate the registry to determine the locations of these SQLDMO files. These two issues made support for both versions of the SQL server engine more trouble than it was worth. Since we knew we could distribute the SQL Server 2000 engine with our application, we decided to drop support for the older engine. The only drawback of the decision is that users of full *SQL Server 7.0* will have to upgrade to full *SQL 2000* in order to support more than five simultaneous users.

Completion of the HFACS Connection component and the required program setup files laid the foundation for the business-logic component development effort. With the questions surrounding SQL Engine version problems, *Visual Basic* limitations, *Access* compatibility issues, and operating system differences all answered, we were free to focus on issues only related to the business-logic component. Specifically, the database schema, placement of the components in some type of architecture, object-oriented design, and efficient data-access methods.

THIS PAGE INTENTIONALLY LEFT BLANK

## **IV. HFACS BUSINESS COMPONENT DEVELOPMENT**

### **A. OVERVIEW**

This chapter provides a detailed description of the design and implementation of the HFACS business logic component. We began its development with determination of overall system architecture via a great deal of research and experimentation in the area of COM components. Once this architecture decision had been made, we refined our conceptual model through interaction diagrams using the UML, prepared class diagrams, and identified stages of spiral development for the rest of our work. Great emphasis was placed on the design of database schema and relationships. In the end, the logic for this component was implemented within classes and *Visual Basic for Applications* modules, and then encapsulated in an *Access 2000* project file. The culmination of this phase was a fully functioning beta of the new HFACS system -- ready for a thorough usability study by an independent testing group.

### **B. ARCHITECTURE**

Our first concern in developing the business logic component for the new HFACS system was to determine the architecture in which we would use it. The architecture decision was extremely important, as it would dictate many aspects of our work. The ease of design, opportunity for code reuse, class design considerations, and scalability would all be directly affected by this decision. We began with an investigation of the pros and cons of each method.

A two-tier solution consists of one or more client applications connecting directly to the *SQL Server*. In this arrangement, the client sends requests directly to the server and the server handles the request by passing information directly back to the client. Client-server workload is manipulated through use of stored procedures and/or client-side SQL text requests. The system can be designed so that the client does most of the labor (called a "fat client" system) or so that the server handles the bulk of the work (called "fat server"). For example, a client could request a copy of an entire table from the server and then when the server provides it in the form of a recordset object, it could sort and manipulate the data in any way required. This type of operation places the vast majority

of the work on the client computer, which is good in terms of server side performance, but is poor with regard to the amount of network traffic it produces. Response times associated with network bandwidth can make this type of operation seem painfully slow. Alternatively, *SQL Server* stored procedures can be used to ask the server to perform various querying operations on behalf of the client and then pass back only the desired information. For example, instead of asking the server to send back an entire table's data, a stored procedure could be used to get data pertaining to just one record. Use of stored procedures in this manner reduces network traffic, but places more burden on the server. An additional concern in the two-tier model is the number of connections needed between client and server. Every client needs at least one connection.

A three-tier or n-tier architecture, on the other hand, is comprised of three or more layers of services. The client and the server are still present and make up two of the layers, but a third layer of architecture exists for the purpose of managing connections and requests between the client(s) and server(s). In general, these middle tiers encapsulate the business logic of an application. Middle tiers are exceptionally well suited for handling requests of multiple servers. This is an important scalability concern. It is very common for departments/organizations to grow and desire to create applications which require simultaneous access to more than one database. Two-tiered solutions would require every client to have a user ID and password for every one of these connections. In a three-tier architecture, however, instead of every client making a separate connection to multiple databases, the middle tier can be designed so that the clients connect to it. Then the middle tier connects to the databases -- with only one connection. Client computers act as if they are connecting to only one server, but via the middle tier, they can connect to multiple servers.

Further complicating the architecture decision was our desire to use purely object-oriented methods of programming. One of our research questions was to determine how the linguistic discontinuity associated with relational databases could be overcome. System architecture is directly related to the answer of this question. This is a very complicated topic. In a two-tier solution, many would argue that a true object oriented design cannot be implemented. Two tier solutions rely upon stored procedures, SQL



syntax, triggers, and views to manipulate data. Each of these presents its own set of limitations on data, which combine to form a somewhat constraining environment. In order to completely overcome the limitations of relational database schema and the aforementioned methods to manipulate them, we believe that no stored procedures or other database server functions should be used. To accomplish this in *Visual Basic*, the client would be required to directly access tables and programmatically perform all data manipulation. Classes would be developed for each table and each instance of a class would *require its own connection to the database*. Additionally, in a two-tier object oriented design, these class objects would need complete copies of all table data, resulting in an enormous network burden.

Alternatively, a three-tier object oriented solution could be implemented to eliminate the network traffic problem. The middle tier could be placed on the same machine or local subnet as the SQL server. This however, would still not resolve the problem associated with each instance of an object creating its own connection to the *SQL Server*. To eliminate this problem, two methods are possible: 1) creating a 4th tier to act as a layer of abstraction for the class instances to interface with the server; 2) utilizing a transaction processing monitor capable of sharing connections. Yet, each of these options presents still another set of unique problems. The 4-tier option would require a huge amount of programming in an area that we have no experience. Additionally, installation programs for the components of these tiers would need to be developed. These installation programs would need the capability to install the components on stand-alone clients running *MSDE*, as well as, true *SQL Servers* -- a very, very complex task. The transaction processing monitor option faces similar installation issues. *Microsoft Transaction Server (MTS)* is the *Microsoft* processing monitor compatible with *SQL Server 2000* and *Visual Basic*. Developing a COM component for (MTS) would require programmatically configuring it for use with *MSDE* and *SQL Server* -- another daunting task.

Clearly a three-tier solution would offer more flexibility in the long run, but our research led us to believe that the programming overhead associated with time and complexity made this avenue prohibitive. Nonetheless, in order to further investigate

how complex this endeavor would really be, and to get a feeling for the benefits it might provide, we developed two prototypes for testing. The first prototype was a three-tier implementation of a COM component for use with MTS. It was designed using *Visual Basic 6.0* as an Active-X DLL. An *Access* data project (.adp) was used as a front-end for the component. The two-tier prototype was created using *Access 2000* & VBA with a direct connection to *SQL Server*. We experimented with the design for six weeks, testing various functionalities as compared to a two-tier solution using a mix of server-side stored procedures and client-side SQL requests. In the end, our predictions were confirmed. The three-tier COM component was much more complex to create and manipulate than the two-tier solution. We were able to successfully use it with MTS on a true *SQL Server* installation, but we were not able to get it to work with MSDE. This is not to imply that it cannot be done with MSDE, only that we could not do it in the time available. These reasons, coupled with the fact that modifications could be made later in the life cycle of HFACS to migrate it into a three-tier solution, led us to the selection of a two-tiered architecture. This decision was not made without careful thought and testing. As will be described in the remainder of this chapter, great lengths were taken in the implementation of our two-tier solution to maximize its ability to be migrated to COM at some point in the future and to optimize it for server-side (fat-server) data manipulation.

### **C. SEQUENCE DIAGRAMS**

With the decision to implement a two-tier solution behind us, we were ready to refine the conceptual model for the component in a fashion similar to that used for the connection component. This involved creation of the Sequence diagrams for the typical course of events for our critical use cases. The sequence diagrams that follow illustrate the actor interactions and the operations initiated by them, as well as, their order.

## 1. Add Factors

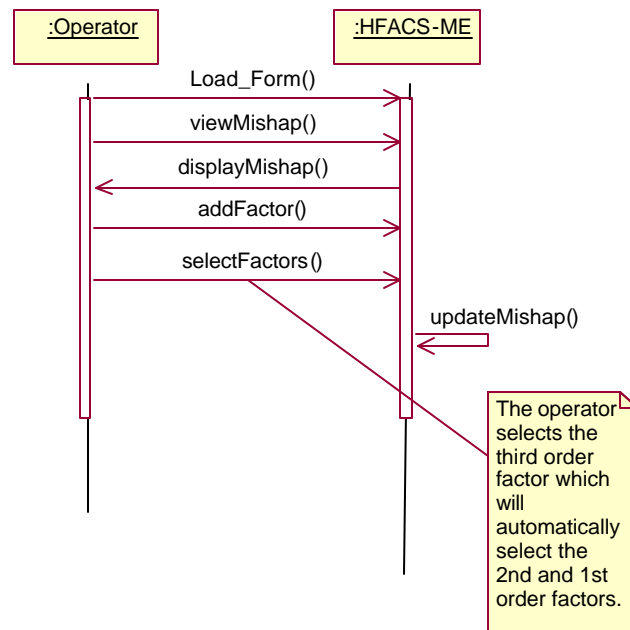


Figure 4.1. Add Factor Sequence Diagram.

## 2. Add Mishap

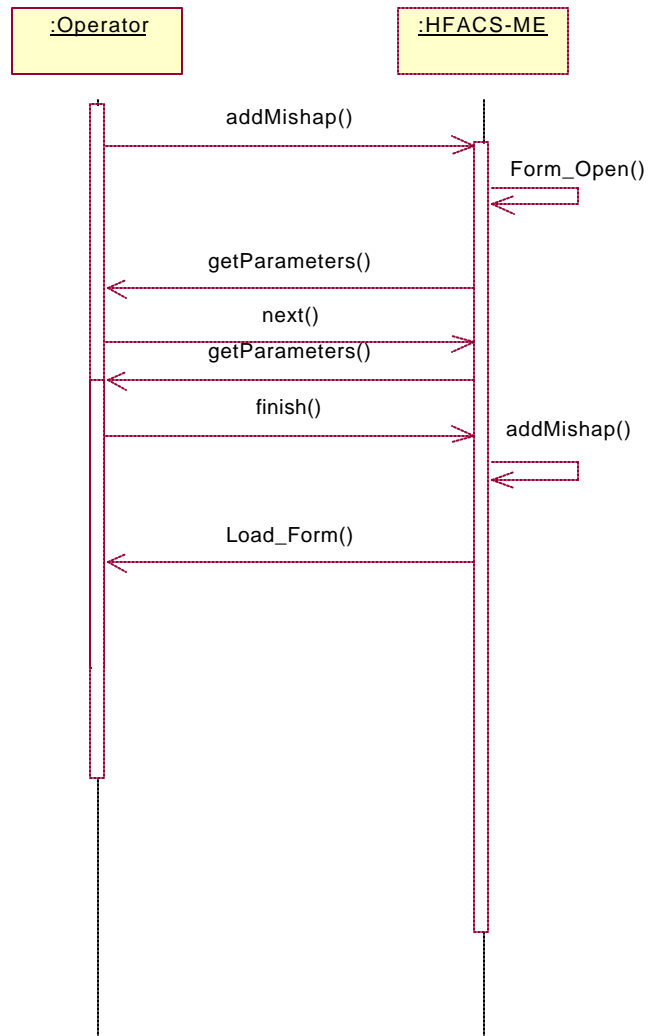


Figure 4.2. Add Mishap Sequence Diagram.

### 3. Graph

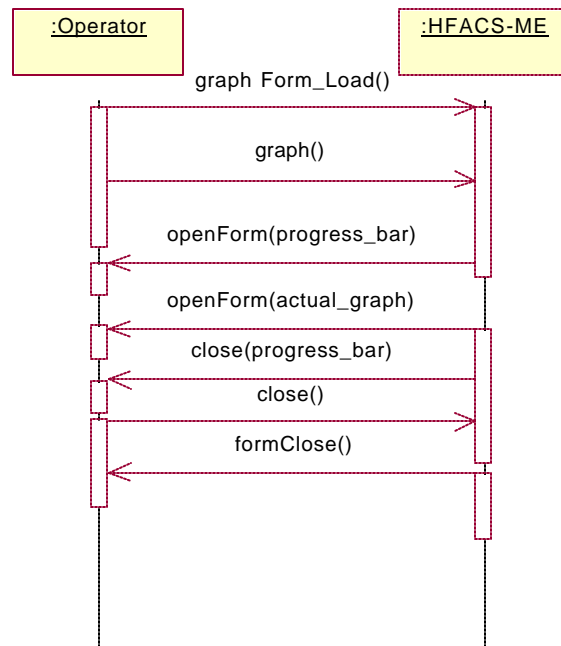


Figure 4.3. Graph Sequence Diagram.

### 4. Edit a Mishap

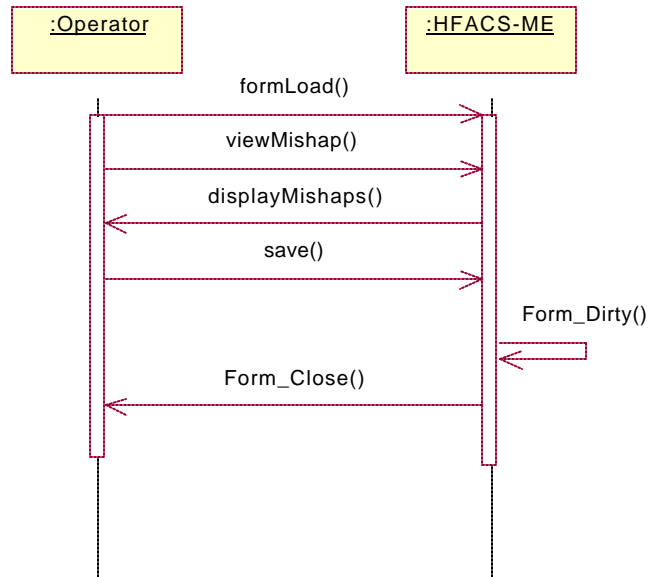


Figure 4.4. Edit a Mishap Sequence Diagram.

## 5. Edit a Factor

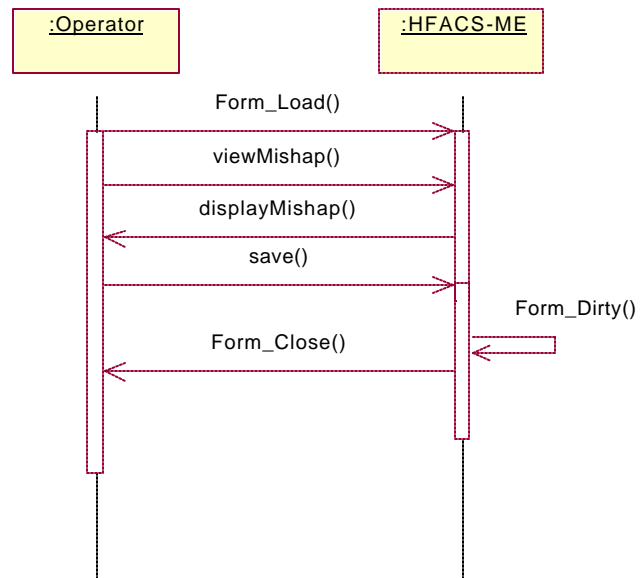


Figure 4.5. Edit a Factor Sequence Diagram.

## 6. Get Summary Report

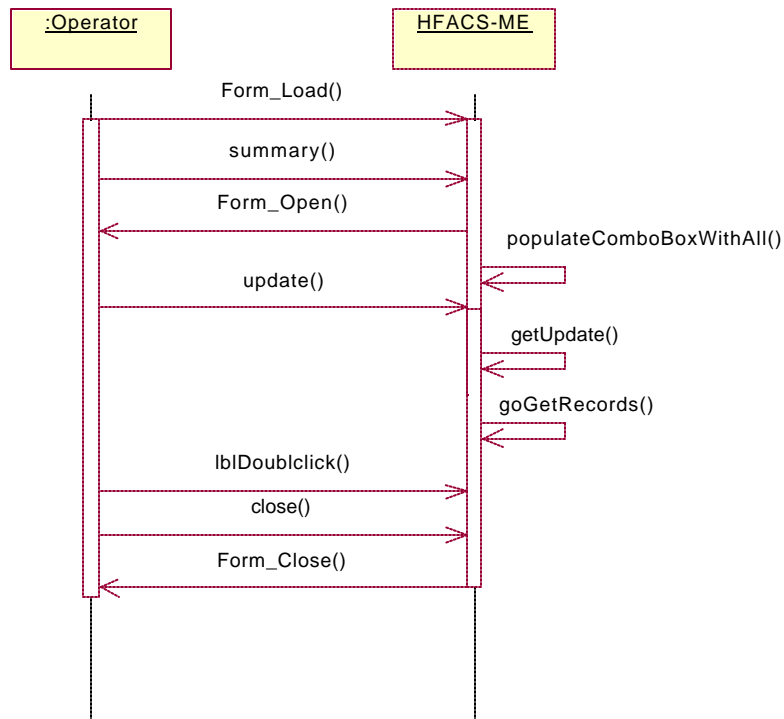


Figure 4.6. Summary Report Sequence Diagram.

## 7. Create a Report

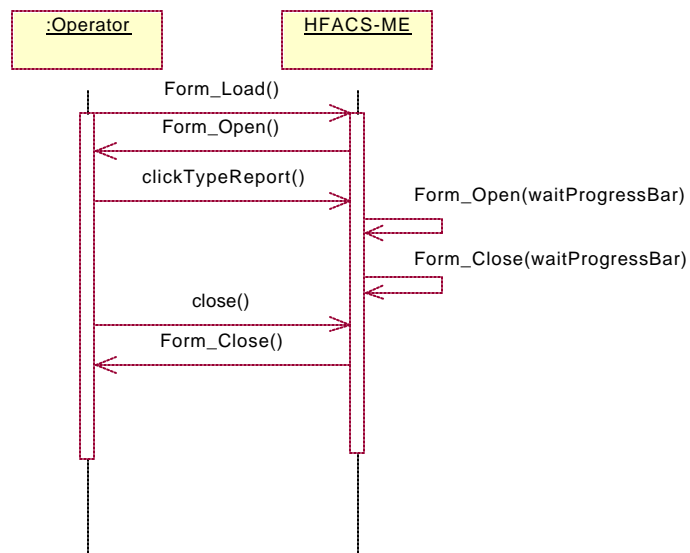


Figure 4.7. Create a Report Sequence Diagram.

## 8. Query

Our requirements specified the ability to query by a single field and by multiple fields. We prepared our use cases to reflect this. During sequence diagram development, we decided to combine these into a single use case by providing the ability to do both operations from the same place.

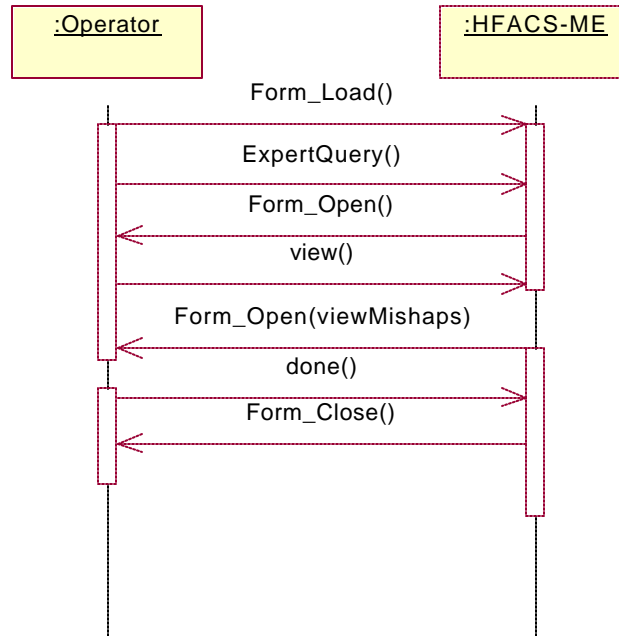


Figure 4.8. Query Sequence Diagram.

## D. COLLABORATION DIAGRAMS

Analysis of our Sequence diagrams allowed us to create Collaboration diagrams to illustrate allocation of responsibilities to objects in the system, specifically demonstrating how they interact via messages. The diagrams that follow provided the level of detail needed isolate the key messaging functions between objects in the component.



## 1. Add Factors

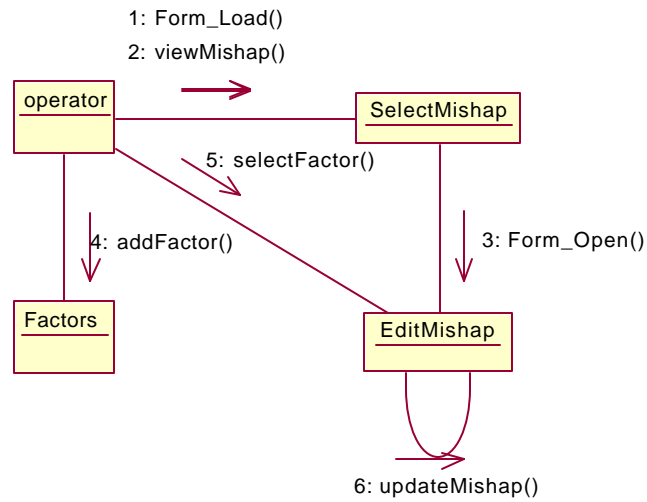


Figure 4.9. Add Factors Collaboration Diagram.

## 2. Add Mishaps

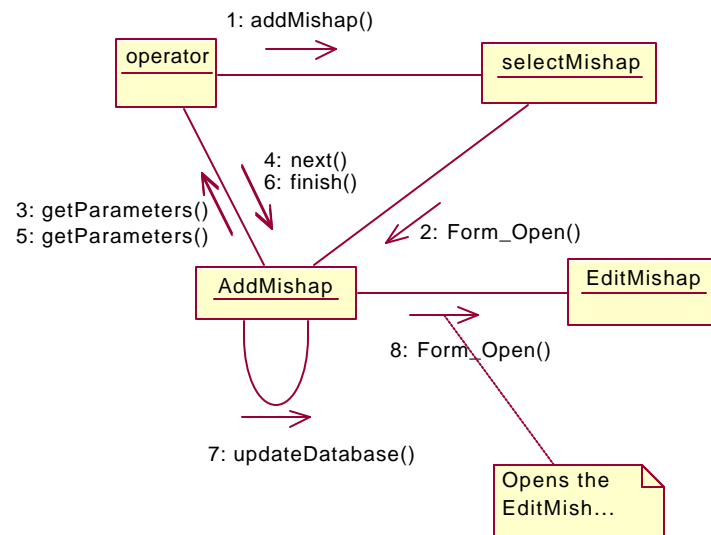


Figure 4.10. Add Mishaps Collaboration Diagram.

### 3. Graph

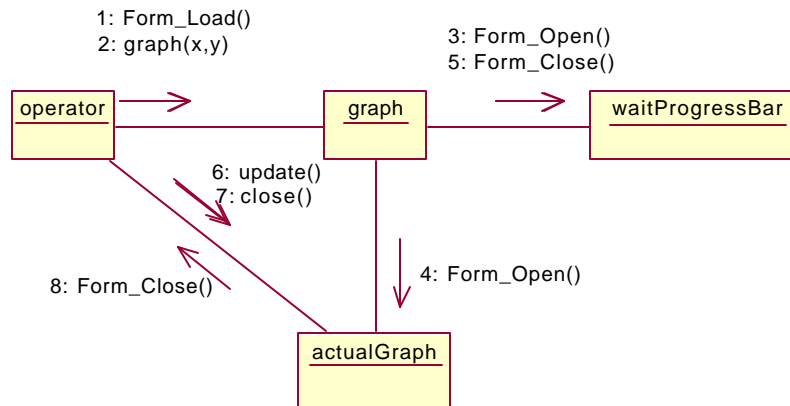


Figure 4.11. Graph Collaboration Diagram.

### 4. Edit a Mishap

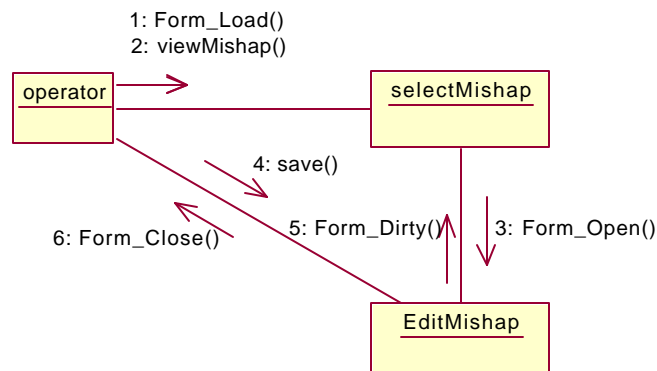


Figure 4.12. Edit a Mishap Collaboration Diagram.

### 5. Edit a Factor

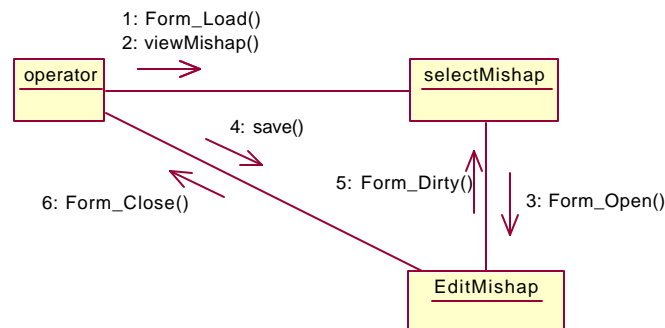


Figure 4.13. Edit a Factor Collaboration Diagram.

## 6. Get Summary Report

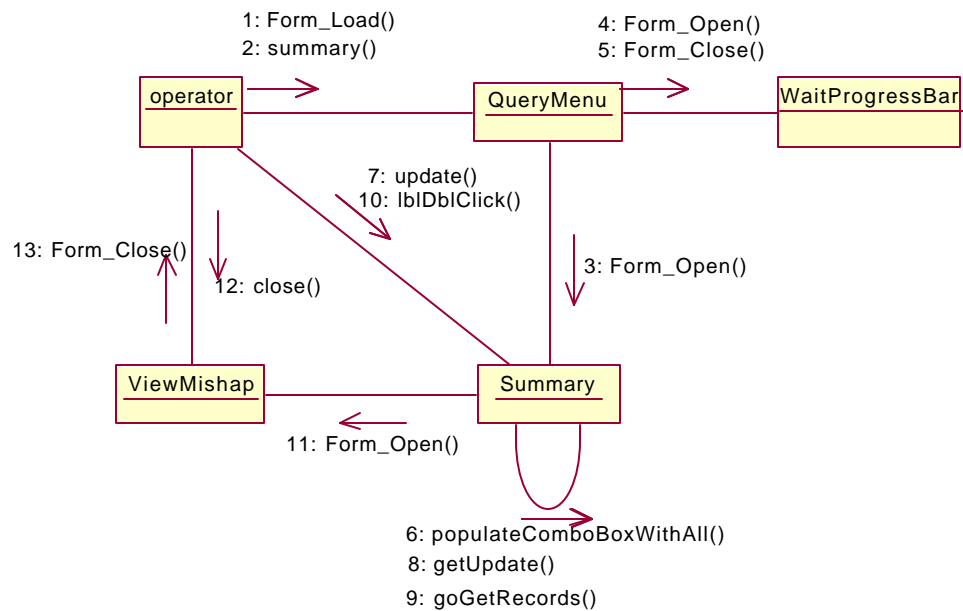


Figure 4.14. Get Summary Report Collaboration Diagram.

## 7. Create a Report

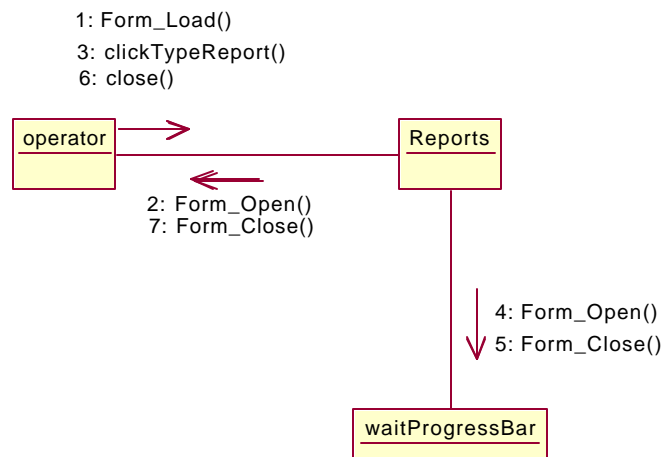


Figure 4.15. Create a Report Collaboration Diagram.

## 8. Query

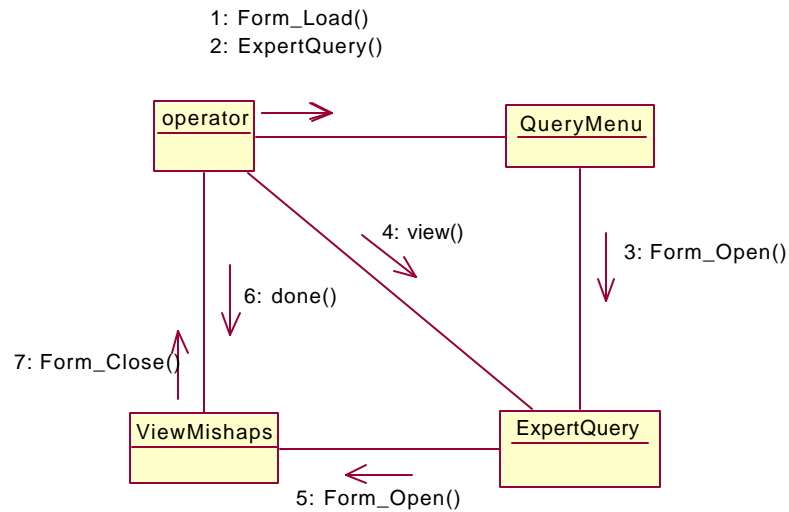


Figure 4.16. Query Collaboration Diagram.

### E. CLASS DIAGRAMS

Finally, the information gleaned from the Collaboration diagrams, empowered us with the knowledge needed to refine our conceptual model. Figure 4.17 illustrates an intermediate level view of the key classes.

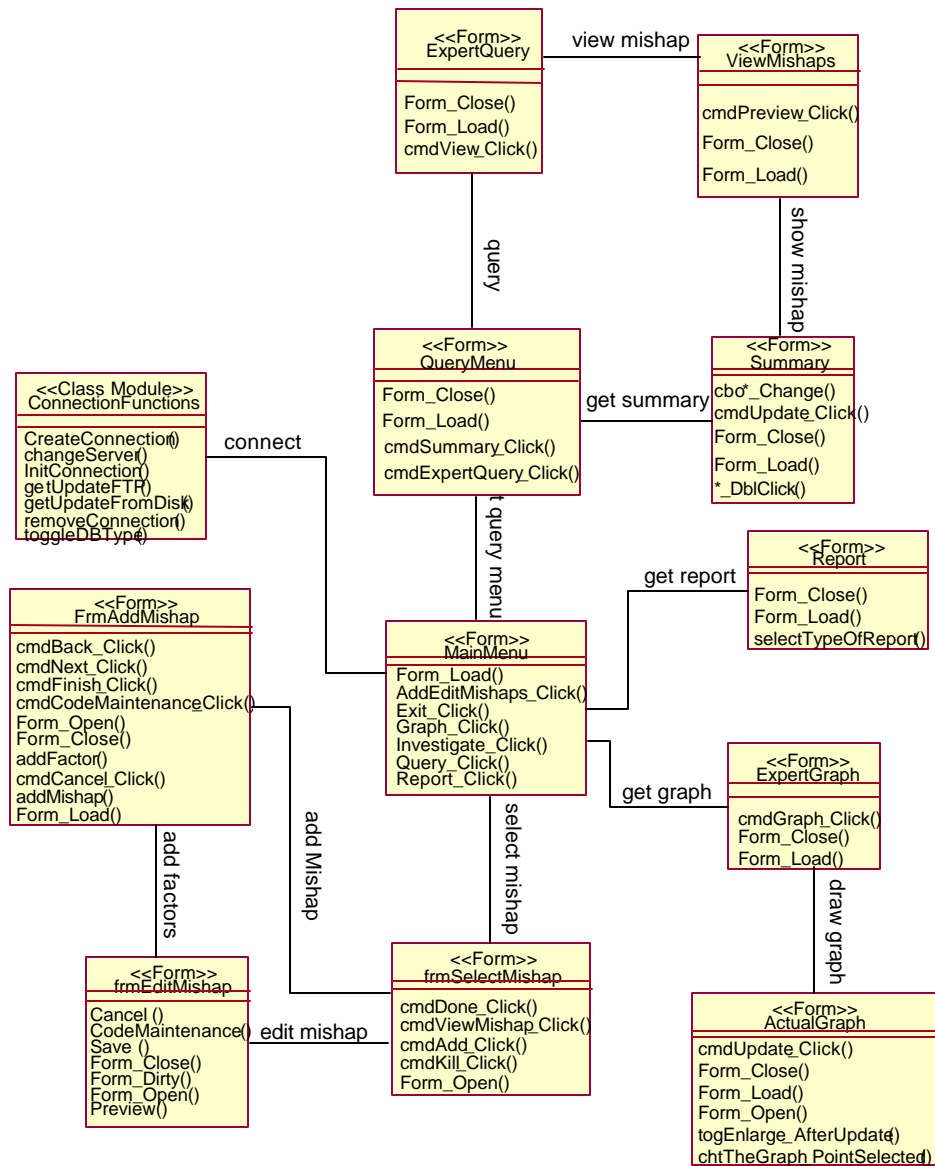


Figure 4.17. Intermediate Class Diagram.

The descriptions that follow provide abridged definitions and explanations for these key classes. They are provided here to document their general functionality in prose format and provide a basis for subsequent discussion of development issues. Detailed HFACS Business component class diagrams illustrating all methods, as well as, complete descriptions of the actual classes the can be found at Appendices D and E, respectively.

## 1. Main Menu Class

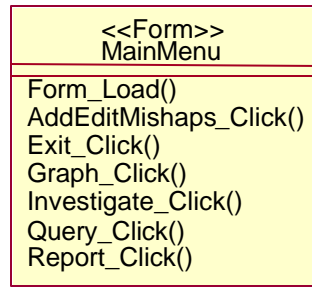


Figure 4.18. Main Menu Class Diagram.

This class is the main switchboard for the program. It is responsible for launching all other processes. It is responsible to launch the add/edit mishap processes, graph process, the investigation process, report process, and the query process. This class will not perform any of these functions but act as a gateway to the other classes.

## 2. Connection Functions Class

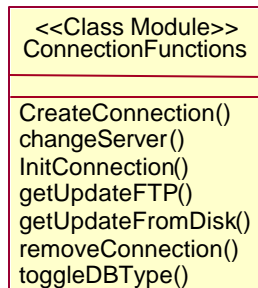


Figure 4.19. Connection Functions Class Diagram.

This class mainly performs the database maintenance and connection to the server functions. It contains the vast majority of the "helper" functions used by the program. It performs the functions for connecting and disconnecting the application to a SQL server, replacing the database via FTP and disk file, and toggling database type from military to civilian and vice versa.

### 3. Select Mishap Class

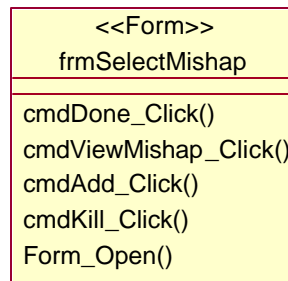


Figure 4.20. Class Diagram for Select Mishap Class.

This class serves the functions to support viewing the mishaps in the database and acts as a gateway to the add mishaps class and edit mishaps class. This class also can perform deletions of mishap records from the database.

### 4. Edit Mishap Class

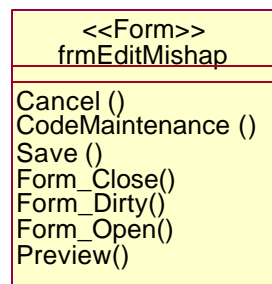


Figure 4.21. Edit Mishap Class Diagram.

This class is used to edit mishaps and add factors. If any changes occur on the existing records, the database is updated to reflect the changes.

### 5. Add Mishap Class

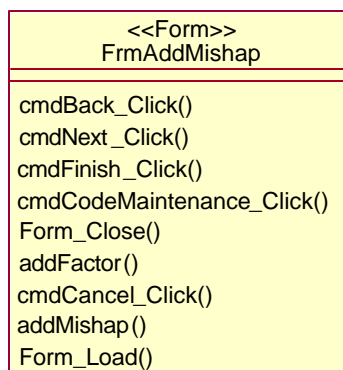


Figure 4.22. Add Mishap Class Diagram.

This class, through series of questions, guides the operator in entering a new record of mishap data into the database. This class will provide guidance and examples when the operator seeks to input the mishap factors that pertain to the new mishap data. Once the operator has inputted all required data, the class will update the database to reflect the new record.

## 6. Expert Graph Class

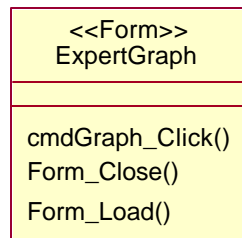


Figure 4.23. Expert Graph Class Diagram.

This class is used to select the X and Y axis criteria and pass the users selections to the Actual Graph class to display the graph.

## 7. Actual Graph Class

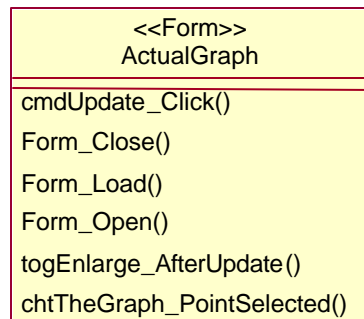


Figure 4.24. Actual Graph Class Diagram.

This class displays the graph with the user selected fields. Initially, graph displayed is the result from the x and y axis values selected by the user in Expert Graph class. Once the graph is displayed, the user can focus the graph into few items such as aircraft type that was involved in the mishaps, or specific location of where the mishaps occurred. The user can also to see the graph of all data (this is the initial view). The user can also choose to enlarge the graph picture.



## 8. Query Menu Class

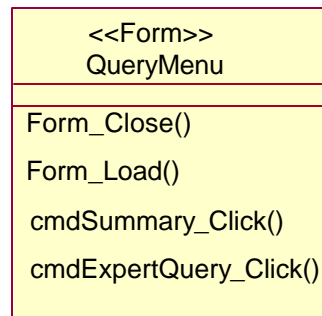


Figure 4.25. Query Menu Class Diagram.

This class acts as a gateway to the expert query class, which will perform query on multiple fields, and the summary class.

## 9. Summary Class

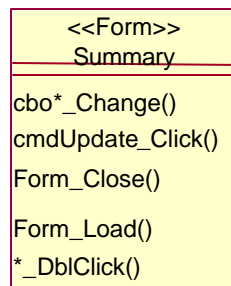


Figure 4.26. Summary Class Diagram.

This class is used to depict the table of factor vs. mishap counts and percentages. It allows the user to select criteria from combo boxes and fills then calculates the values for the table when the user clicks update. When the user double clicks a label in the table, View Mishaps class is launched which will display the mishaps that comprise the data for the label in the summary data display.

## 10. Expert Query Class

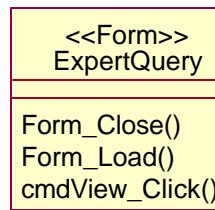


Figure 4.27. Expert Query Class Diagram.

This form allows the user to choose multiple criteria from a series of combo boxes and then query the database to open the View Mishaps class and display the mishaps and factors. When the user clicks "View", View Mishaps class is launched which will displays the mishaps that matches the criteria established in the user selected combo boxes.

## 11. View Mishaps Class

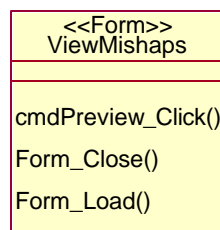


Figure 4.28. View Mishaps Class Diagram.

This class displays the mishap data responding from the Summary class and the Expert Query class. The data displayed is not editable because it has read only functionality.

## 12. Report Class

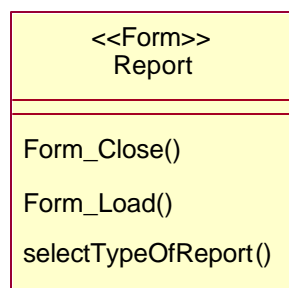


Figure 4.29. Report Class Diagram.

This class is the form for selecting the type of report to run. The class will display the results that corresponds to the user's parameter that was selected in a combo box. It basically performs the functionality of sorting the data. For example, if the user selects the report and the parameter selected is by year, then the data will be created in the report and the data will be sorted by year.

#### **F. IDENTIFICATION OF SDM STAGES**

As discussed in Chapter 3, we utilized the Spiral Development Method (SDM) as a guide to control risk throughout the component development process. During the process of developing the class diagrams, several key issues arose which led us to our choices for SDM design stages. Foremost of these issues was development of the database schema. The old versions of HFACS (military and civilian) used different schema that were not compatible in any way. In addition, we needed to coordinate with other groups working on HFACS to develop an integrated solution that would meet everyone's needs. We knew that in order to have data to work with while developing the query and graphing classes, the ability to add, edit, and delete data would need to be done first. Similarly, security concerns needed to be ironed out prior to coding the add, edit, and delete classes. Based on these observations we identified the following five stages of cyclical development:

- Stage 1 - Creation of Database Schema
- Stage 2 - Analysis of security
- Stage 3 - Creation of add, edit, and delete classes
- Stage 4 - Query, Graph, Reports
- Stage 5 - Test

#### **G. IMPLEMENTATION - STAGE 1**

One of the goals of our design was to overhaul the database schema in the old military and civilian HFACS systems such that a common application interface could be developed for both types of data. This would make maintenance of the application code much easier because, in effect, both databases would actually be one database. At the same time, however, other groups working on the project had their own design considerations to contend with. We began the process by refining field names for data to generic terms compatible with both versions. For example, instead of using fieldnames

like "Service" for the military version and "Carrier" for commercial version, we decided to use "Organization" -- which would apply to both types of data. Numerous changes of this nature were made.

Our next step was to develop relationships to define the data. Structured Query Language and relational databases are generally restrictive by nature. In any complex project, developers face the limitations imposed by relationships every day. This is generally a result of the normalization and other "structural rigidities" of relational data. Because of these restrictions, we took great care when defining the structure of the database tables.

Normalization consists of the standard rules of predicate calculus applied to relationships to prevent a design that can cause repeated and inconsistent data. Poorly designed relationships gives rise to complex SQL statements, with multiple joins, necessary to re-mold the structure. We began by reviewing the following standard definitions of 1st, 2nd, and 3rd normal forms and applying them to our proposed table definitions [Ref. 30]:

- First Normal Form - Removes all repeating groups of data by giving each logical group a separate table and providing a primary key in each.
- Second Normal Form - Key fields are chosen so that non-key fields depend on all fields in the primary key.
- Third Normal Form - No fields depend on other non-key fields.

The products of our review consisted of tables for the entire database in 3rd normal form. Figure 4.29 illustrates our tables in 3rd normal form. At this point, we entered the data into *Microsoft Access* using a *JET* engine and conducted experiments to determine how we could manipulate keys and relationships to provide the fastest performance.

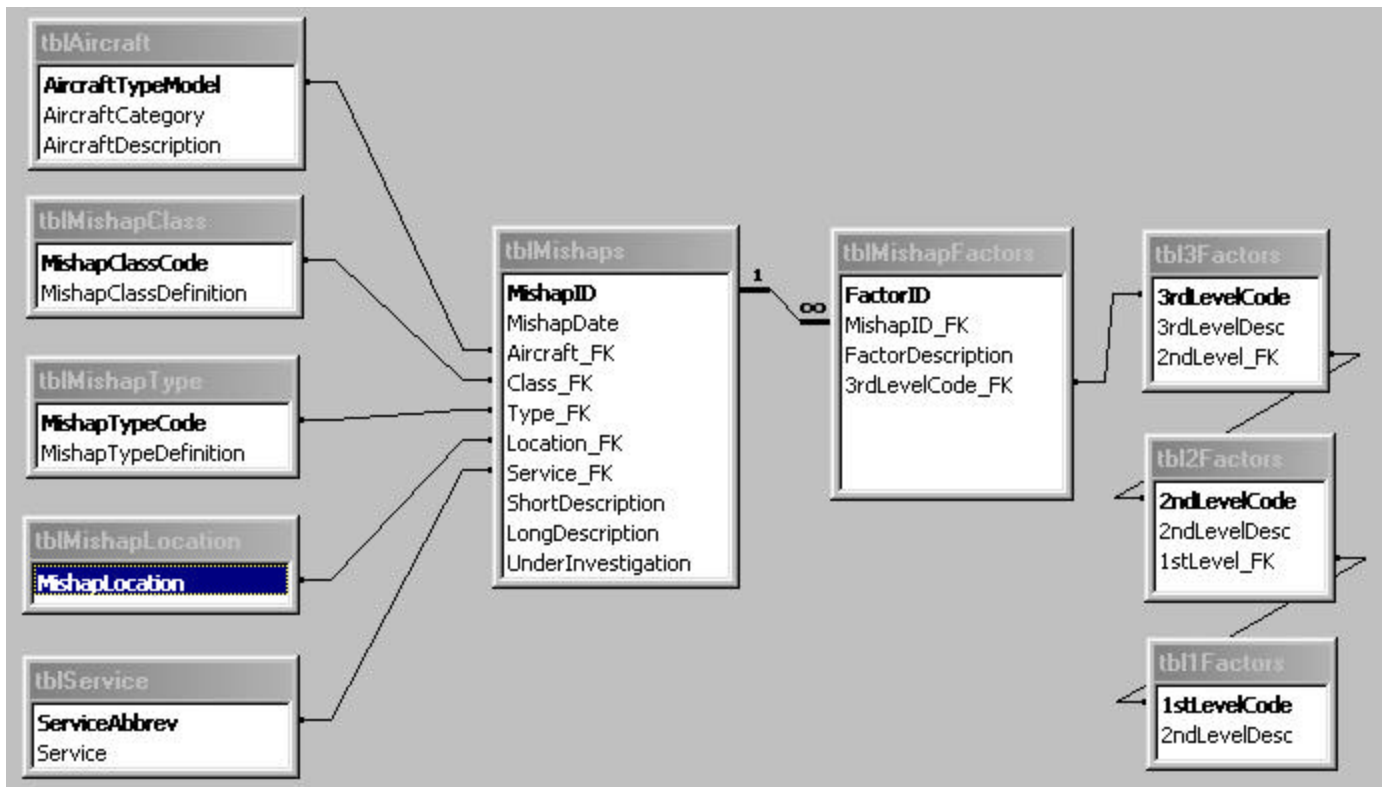


Figure 4.30. HFACS Tables - 3rd Normal Form.

In general we decided to adopt third normal form for all tables with one exception -- the "Factors" table. The "Factors" table consists of first, second, and third level factors, which define the HFACS-ME taxonomy. In third normal form, each factor gets its own table with its own primary key. In reality, however, the third, second, and first level factor *combinations* are each uniquely defined by the third level factor. Since the HFACS taxonomy is well defined, it is highly unlikely that many new factors will be added. This creates a situation where the factors are referred to merely for "lookup" purposes. For performance reasons, we decided to place all factors in single table. Similarly, in defining relationships, we chose to enforce referential integrity for cascading deletes only between the Mishaps and Factors tables. To accommodate both types of database (civilian and military), we added a table for "Database Type" and placed foreign key fields in the tables where differentiation of data would be necessary. This made it possible to select the data for the appropriate database type with only a single extra join per query. Our final agreed upon solution is illustrated in Figure 4.30.

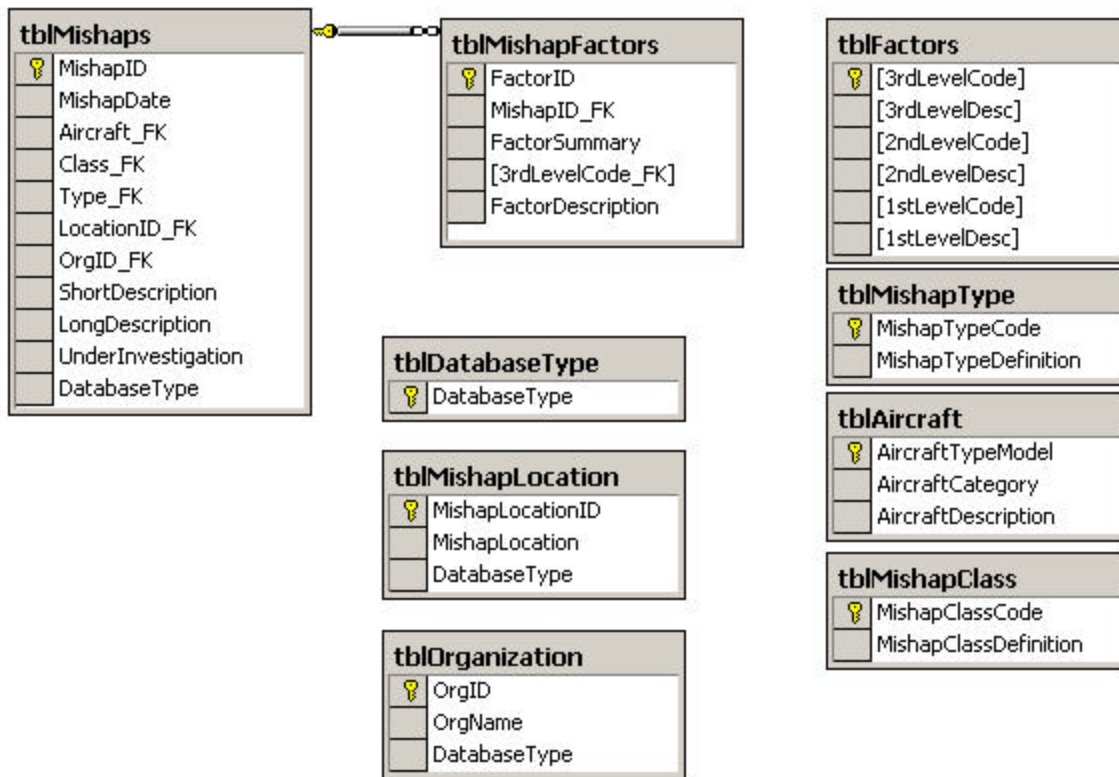


Figure 4.31. HFACS Tables - Final Solution.

Throughout our development, we created many very complex queries to display data in various formats for graphing and advanced queries. Our conscious decision not to blindly adopt third normal form did not appear to hinder us in any way.

## H. IMPLEMENTATION - STAGE 2

Without a doubt, the security aspects of this project were the most difficult to implement. To start with, we have the three modes of security used with *SQL Server*: 1) NT authentication mode, 3) *SQL Server* mode, and 3) mixed mode. Compounding this complexity were NTFS file and share permissions associated with *Windows 2000* and *Windows NT*, as well as, the complete lack of any intrinsic security in *Windows 95/98/ME*. Our requirements stated "access to the add/edit feature of the database must be controlled via a password mechanism." Follow on discussions with our sponsors

further refined this statement to mean that unauthorized users should not be able to add or edit the official data in the database. Unauthorized users are further defined, as "anyone not specifically identified by our sponsor as authorized to make changes." This creates a fundamental problem. A *SQL Server* database file that is physically distributed to an organization can be attached and modified by anyone with *SQL Server* administrative privileges. This means that the only method to ensure unauthorized access is to maintain a single copy of the database in one physically secured location. Users could then be granted access using one of the modes of *SQL Server* security. Stated differently, the only way to secure this type of database is *not* to physically distribute it.

Since this was not feasible due to the requirement for connectionless operation, the next best alternative was to make it difficult to change the data. To accomplish this, NTFS permissions and *SQL Server* permissions were not enough, as *Windows 95/98/ME* users with a default installation of MSDE using the "sa" login and a blank password (the system defaults) can change the data in the database. The solution was to create a third level of security within HFACS -- its own security module. To do this, a data store of user IDs and Passwords must be accessible during logon to validate the users as administrators. There are several complex methods available for this type of implementation. Our recommendation for future efforts would be to utilize a secure key server. For our purposes, however, we chose to store the passwords in a hidden table in a separate *JET* based *Access* file. This separate database can be password protected and the password can be hard-coded into the compiled HFACS application to allow it access to the data without giving users the ability to see the password. The obvious problem with this is that if the password is compromised, there is no way to change it. A key server would solve this dilemma, as it would provide a single point for password validation and passwords could be changed if they were compromised. It would also, however, require network access of some kind.

In the end, our security arrangement is good enough to keep people from accidentally accessing the add/edit features of the database. A determined person with malicious intent could gain access and change the data. As it stands, the most complex security is provided by the *Windows 2000* platform. In order for *Windows 2000* users to

access the administrative features of the database, they must be a *Windows 2000* system administrator, a *SQL* server administrator, and an HFACS administrator. All these checks are programmatically verified by our business logic component every time a user attempts to access the administrative features of the database. On *Windows 95/98/ME* with a default installation of MSDE, however, there is only one security check -- for HFACS administration privileges.

## **I. IMPLEMENTATION - STAGE 3**

The first classes that we implemented in code were the add, edit, and delete classes. This was the logical progression for our SDM as these functions needed to be operable before we could build the query and graph classes. As described previously, the add, edit, and delete classes posed special security problems. Adding to the complexity was the requirement for users to initiate an "investigation" of a new mishap. Investigation of a new mishap is really the same operation as adding records to the mishap and factors tables, but we could not allow "normal" users database "write" access to these tables -- only an Administrator is authorized to add mishaps to the official database. In addition, if a normal user were able to add records to the database, any database replacement operation (via FTP or disk as described in Chapter 3) would overwrite their input. A method was needed to input persistent data that would not be overwritten by a database update (replacement operation).

The solution to this problem came in the form of a separate *Access* database. Each HFACS installation includes a separate *JET* based database (*Investigate.mdb*) that provides the local functionality for adding, editing, and deleting mishaps for investigation purposes.

By implementing our solution in this fashion, all users can initiate an investigation without altering the official data in the *SQL Server* database. Additionally, a database replacement operation will not overwrite their saved investigations. The *Investigate.mdb* module uses the same program logic as the *SQL Server* version of the database and can be launched from within the main HFACS application without the user even realizing that they are using an entirely separate program. This solution has the added benefit of the capability to operate as its own standalone program. This is



pertinent because in the event of the implementation of a key server solution at some point in the future, users disconnected from the Internet will still have the capability to initiate investigations -- an interesting and viable option to alleviate some of the security concerns.

Two other interesting discoveries were made during this phase. First, the Investigate.mdb database had to have a capability to determine what type of HFACS implementation launched it. For example, if the user of the civilian configuration of the database launched the investigation module, then the investigation module needed to run with civilian options specified for inputting and editing data. This proved a somewhat difficult problem, as we desired to maintain the ability for the Investigate.mdb program to operate in standalone mode. For several days we experimented with command line arguments without success. The solution was to add the "iniFile" class from the HFACS connection component to this standalone database. In this manner, the Investigate.mdb program is capable of reading from the HFACS.ini file to determine what mode to open in. If the HFACS.ini file is not present, the default setting of "military" is used. This served as a testament to the code reusability of our project to this point.

A second and unrelated, yet interesting discovery, dealt with the usability of the general HFACS graphical interface. A limitation of *Visual Basic* and *Access* is its inability to automatically resize controls on a form (in a class) when the user stretches a window. This may seem trivial, but in a data aware application that uses grid controls, anyone using a monitor with resolution of 1024 X 768 or greater is stuck looking at a small box in the middle of the screen that is barely legible. This problem quickly made itself apparent during the development of the add/edit and delete classes. A search of the Internet yielded the code for a form resizing class from *Database Creations, Incorporated*. Utilizing this class we were able to provide support for dynamic resizing of forms based on the user's screen resolution, greatly enhancing the usability of our program.

## J. IMPLEMENTATION - STAGE 4

The queries, graphs, and reports classes were the final code development effort for our implementation. The query classes were very straightforward and posed no significant problems. Graphing and reports on the other hand, were a special challenge.

I'll describe the graphing issues first. As alluded to in Chapter 2, Access used to support only one database engine -- *JET*. When support was added for *SQL Server*, many of the RAD features associated with *Access* were not supported. Graphing is one such example. *Access* does not provide the capability to pass input parameters to a bound graph control when *SQL Server* is used as the data engine for an *Access* project. To circumvent this problem, a *Visual Basic Active-X* control was utilized. *Visual Basic 6.0 Enterprise Edition* provides a redistributable *Active-X* chart library called *MSChart20.ocx*. By including this library as a reference within *Access*, all the methods and properties became available and an impressive set of charting options presented itself. This control is not for the weak at heart. It is the largest *Active-X* control we have ever used and documentation for it is confusing. To populate a chart requires creation of a separate "datagrid" object. In order to use it you must programmatically define every position in the grid including labels, font sizes, orientations, styles, and so on. Further complicating matters, this *Active-X* add-in does not fully support *Access*. It was designed for true *Visual Basic* and not VBA. *Access* provides no means to manipulate the *Windows* clipboard objects, which are integral for sending images of MS Chart graphs to an attached printer. To circumvent this, another dynamic link library was developed using pure *Visual Basic*. Its sole purpose is to provide explicit print support to HFACS for the *Windows* clipboard (*HFACSClipboard.dll*).

Just after getting the charts to work with sample data, we made our next unforeseen discovery -- *SQL Server* does not provide embedded support for cross-tab queries. A cross-tab query is a spreadsheet-like summary of the things specified by the row header and column headers that is created from a table or query -- but, only when using the *JET* database engine. This type of query presents summary data in a spreadsheet-like format created from the fields that you specify. In this specialized query, row and column totals can be generated on the fly. For example if we wanted to

create a query that displays the *type of aircraft* field as the row heading and the *third level mishap factors* as column headings, with each cell containing the total count of mishaps for each type of aircraft with that factor, we could do it. Figure 4.31 illustrates some sample output of such a query.

	Aircraft_FK	ADA	ATT	COM
	A4	0	2	0
	AV8	0	12	9
	C12	0	1	0
	C130	0	4	2
	C2	0	0	1
	C9	0	1	1
	E2	0	5	4

Figure 4.32. Example Crosstab Query.

This kind of query is ideal for populating datagrids for MS Chart controls, as well as, for tabulating reports. Unfortunately, and as unbelievable as we found this to be, *SQL Server* cannot create these types of queries. Weeks were spent trying to circumvent this problem. Luckily, we discovered a R(eplacement) for the A(ccess) C(ross-tab) query. RAC is an application that runs on *SQL Server* and produces two-dimensional cross-tab reports. It was designed by *Steve Dassin* and was included in HFACS with his permission [Ref. 31]. RAC has various options that make it possible to enhance the traditional *Access-JET* cross-tab functionality by providing additional capabilities over those in *Access*. RAC has a number of report like format capabilities that enhance the appearance of table data. In addition to producing cross-tab reports, RAC can be used to transpose fields, split delimited strings and create delimited strings. RAC is written in transact-SQL exclusively for *SQL Server* version 7.0 and above. A set oriented approach is employed in most places and RAC does NOT use any cursors. RAC can accommodate any level of server and was so easy to use that we were able to create cross-tabs and reports with it in minutes. We cannot thank Steve Dassin enough for this contribution. Prior to implementing RAC, our graph and report queries were so complex that they took 3 - 5 minutes to return a result on a dual-processor Pentium III 550 server.

Our last great challenge dealt with report generation. We feel that *Access* has never offered seamless support for reports. Even today the entire report generation interface in *Access* is noticeably disconnected from the rest of the program. Projects utilizing a *SQL Server* engine compound the adverse effects of this discontinuity. In order to create the types of reports we desired, support for specifying control data sources (mapping textboxes to table fields) needed to be assigned at runtime -- after the program was compiled. This proved extremely troublesome and we were never able to get it to work properly. On forms, this type of runtime change is simple. Instead, we fell back on the power of Steve Dassin's RAC and some extremely complicated transact-SQL to generate the desired output.

## **K. IMPLEMENTATION - STAGE 5**

Testing the newly created HFACS business logic component and its related components (*Investigate.mdb* and *HFACSClipboard.dll*) began with small-scale tests on the *Windows 2000* and *Windows 98* platforms.

### **1. Windows 98 Tests**

On systems with full *Access 2000* or newer installed, running *Office Service Release 1/1a*, we found no deficiencies. The same was true of systems that utilized the runtime version of *Access* provided by our program. We did make some interesting discoveries in terms of usability, however.

First, we discovered an issue with how *Windows 98* configurations connect to other *Windows 98* machines on a network. When we tried to connect a *Windows 98* computer running HFACS to another *Windows 98* computer running HFACS, it would not work. But, in a similar configuration on the *Windows 2000* O/S, it would work. As it turns out, this behavior is by design. *Windows 2000* will default to a network connection between client and server on the same network subnet using Named Pipes to connect. This requires no additional configuration. *Windows 95/98/ME* computers, however, do not support Named Pipes. For this reason, TCP/IP connections must be used. A TCP/IP connection requires a system Data Source Name (DSN) to be built. Once a system DSN was built, we had no problem connecting. See the *Windows 95/98/ME* help documentation for detailed instructions on how to build a system DSN.

Second, we found that print preview support for reports in *Access* requires that a default printer be installed. We realized that to actually print a report required a printer, but we had not realized that *previewing* a report required one. This is caused by the requirement for printer specific data in order for *Access* to generate a *What You See Is What You Get* (WYSIWYG) preview. We added some error handling to prevent the runtime version of *Access* from crashing when users without a printer attempt to preview reports. Incidentally, a printer does not actually have to be connected, just installed. We "tricked" several of our *Windows* computers by installing printer drivers for printers that really did not exist -- previewing reports worked fine.

## **2. Windows 2000 Tests**

*Windows 2000* installations exhibited the same problems as *Window 9X* systems in terms of report compatibility with a default printer. In general, *Windows 2000* installs proved significantly more difficult than those on *Windows 9X* -- the NTFS file system was extremely troublesome to manipulate. Our application would only install to the profile of the *administrator* performing the installation. Through much experimentation, we determined that to configure HFACS for use by other than system administrators required the following steps:

- Using the same administrator account that was used to perform the HFACS installation, the program must be run for the first time by clicking Start -> Programs -> HFACS-ME. When HFACS runs for the first time it performs the actual installation of the HFACS database by attaching it to the SQL server engine that is running on the same machine. HFACS must be successfully connected to the SQL engine at least one time using an *administrator* account before any further configuration is attempted. A successful logon indicates that the database was properly attached to the SQL server engine and that it can be shared for use by others. If this step is not performed prior to giving users with other than administrator rights access to the program, they will not be able to launch the program as they will not have sufficient permissions to attach the database data files (hfacs.mdf & hfacs\_log.ldf) to the engine.
- Next, a copy of the folder containing the shortcut to the HFACS-ME program must be pasted from the administrator profile to the *All Users* profile. This places a program group on the start menu for all users of the machine.
- Finally, file permissions for all users that will require access to HFACS must be assigned to the HFACS program directory and the *Visual Basic*

virtual machine library. Assuming a default installation and a normal domain structure these files are located in the following directories:

- C:\Program Files\HFACS (Give Modify permissions to Domain Users for the entire subdirectory).
- C:\Winnt\System32\msvbvm60.dll (Give Everyone permissions to Read & Execute just this file).

The actual permissions will vary from computer to computer and domain to domain, depending on the configuration settings of the LAN. Additionally, on computers running *Windows 2000 Professional* that have *Visual Basic 6.0* installed, users should be made members of the "Power Users" built-in group in order to access HFACS.

## V. CONCLUSIONS AND RECOMMENDATIONS

### A. CONCLUSIONS

A well designed object-oriented system is one in which responsibilities are allocated to classes of objects. Proper partitioning of these objects dictates a well thought out distribution of responsibilities among subsystems. This type of system is easier to develop, simpler to enhance, and more flexible than traditional procedural code. This thesis described meticulous methods of software reengineering throughout the HFACS development process in order to capitalize on the benefits that this type of object-oriented methodology has to offer. Through our 11-month research effort, we have come to the following conclusions with respect to our research questions:

**How can a *Microsoft Access* based implementation provide multi-user access to the same database in a client-server environment while ensuring the ability to scale to a large number (potentially thousands) of users?**

Our experiments with the JET and the *SQL Server* data engines clearly demonstrated that JET was not capable of true multi-user access for more than a handful of simultaneous users. The JET engine is merely a file-server and cannot perform server-side data manipulation. The only way that *Access 2000* could provide the scalability capable of meeting our requirements was to use it in the role of a client "shell" in conjunction with a data engine other than JET. In this manner, the functionality of a robust data engine capable of scaling to large numbers of simultaneous users, with support for replication, server-side querying, and automation could be implemented. Our review of commercial products demonstrated that several databases offer this type of functionality, but our desire to keep our solution *Microsoft* based, as well as, *SQL Server's* royalty free distribution policy for the *Microsoft Data Engine / SQL Server Desktop Engine*, made it the logical choice. Use of the SQL data engine solved the problem of multi-user, client-server, development when using *Access* as a client.

**How can the linguistic discontinuity associated with object-oriented concepts and relational databases be overcome when limited by requirements to use certain types of software implementations (e.g. a *Microsoft Access* based solution)?**

Linguistic discontinuity refers to the procedural style limitations associated with ANSI SQL and relational databases in general. As discussed in Chapter 4, system architecture is directly related to the answer of this question. In order to completely overcome the limitations of relational database schema, we believe that no stored procedures or other database server functions (e.g., triggers, views, proprietary engine capabilities, etc.), should be used. To facilitate this, some type of software middle-tier must be developed. This creates many complex scenarios associated with the number of connections between instances of objects and, as an organization grows, with the number of databases that clients have to connect to. Additionally, the amount of knowledge needed to "add-in" a middle-tier of software associated with a specific vendor's product can prove to be immense. We found that use of *Microsoft Access* was not the issue when making our decisions related to system architecture for overcoming this intrinsic linguistic discontinuity. Instead, the most significant issue was related to our requirement for HFACS to operate as a stand-alone program using its own data engine in a non-networked environment.

In our experiments, we were unable to effectively use a transaction monitor (MTS) with MSDE. This proved prohibitive in terms of a three or more level design. Without the ability to use MTS with MSDE, we could not programmatically create an environment where HFACS was able to connect to the local instance of the database engine through a completely encapsulated and totally object-oriented business logic component. We were, however, able to successfully implement a prototype business logic component with *Access*, *MTS*, and the *Enterprise Edition* of *SQL Server*. This demonstrated that if the need arises for HFACS to be migrated to a three-tier architecture in support of large enterprise level operation in the future, a business logic component for the middle-tier can be created to do it. Armed with this knowledge, we constructed our two-tier solution to maximize its ability to be migrated to COM or DCOM supporting a middle-tier of business logic at some point in the future. This was accomplished by



placing great emphasis on object-oriented methods using classes, VBA modules, and use of unbound controls in our *Access* based business logic component. Use of unbound controls greatly increases opportunity for future code reuse by eliminating reuse issues associated with directly linking "visually designed" controls on *Access* forms to tables in the data engine.

**The current military and civilian systems provide similar functionality, but use different database schema. How can a common interface be developed for both types of data?**

As it turns out, we were able to easily identify changes in the taxonomy of our tables to make schema conventions applicable to both civilian and military data. The data manipulation is the same. In the future, this will most likely not be the case. The life of a database is really comprised of many small development cycles. If the need arises to design new data structures, our HFACS system can be migrated along two separate development paths. Until that time, however, our solution provides a single code base. This removes layers of complexity in terms of maintenance and design time. We strongly recommend that any follow on development cycles continue to implement methods which harmonize both versions of the program into a single code base for as long as is feasible. To summarize, the solution to this research question was not to create a common interface, but to create common data schema.

**How should database schema be changed to provide the best performance, scalability, and opportunity for code re-use?**

In conjunction with our changes to streamline both military and civilian data into common fields, we made two decisions that we feel improved the performance of our database, without adverse impact on the ability to be scaled.

First, our decision not to enforce referential integrity (cascading deletes) except in one instance makes our database much easier to adapt to a completely object-oriented three-tier architecture later in its life cycle. What we are specifically referring to here are the problems that cascading deletes can cause if, in the future, it is desired to manipulate the HFACS database to completely object oriented code (without stored procedures,

triggers, etc.). Relationships enforcing cascading deletes can limit attempts to create a completely object-oriented middle tier while continuing to provide simultaneous support for older, two-tier versions of the program. This will most likely be the case. We believe that if an implementation of this type is ever developed, it will no doubt only be used on machines running true *SQL Server*. Computers running MSDE will probably still rely on the two-tier architecture as a result of the MTS add-in problems previously described. Whatever the case, any new development of this nature will be an extremely complex programming initiative. Our conscious effort to limit cascading deletes to the single *Mishaps-Factors* relationship should help ease that burden.

The second schema change we believe improved the performance of HFACS was to incorporate all three levels of mishap factors into a single lookup table. As described in Chapter 4, there are only 33 third level factors. Each third level factor really defines the second and third level factors. The decision to treat these relationships as a single lookup table made queries less complex in the vast majority of our stored procedures.

**In the past, *Microsoft* has deployed new versions of *Microsoft Access* and *Visual Basic* that were not (fully) backwards compatible with previous versions. This caused great discontent among users of applications designed to run under the older versions of these programs. How can our systems be designed to isolate them from problems associated with new versions of *Microsoft Access*? Specifically, the pending release of *Microsoft Office XP*, *Microsoft Office 2002* and *Microsoft Visual Basic.NET*?**

We feel that second to our investigation into use of the *SQL Server 2000* engine to house our data, this was the most important area of our research. As evident from the incompatibilities we found in attempts to upsize *Access 97* databases to *SQL Server 7.0* and subsequent attempts to migrate from the *SQL Server 7.0* format to that of *SQL Server 2000*, changes in *Microsoft* technologies are the greatest threat to the continued operation of our program. For these reasons we attempted to utilize as many non version-specific aspects of *Access* that we could.

To begin with, we chose data access technologies, like ADO, that *Microsoft* recommends to help ensure future product compatibility. In fact, our HFACS connection component first tries to make its connections using SQLDMO and if this fails, it switches to ADO. This redundancy greatly improves its ability to operate in several different environments. Next, we implemented programming methods that *Microsoft* recommended for compatibility with the next generation of *Visual Basic* (VB.NET). Third, we invested in the *Developer* editions of *Microsoft Office* which allow royalty free distribution of *runtime Access*. In this manner, even three years from now, when the vast majority of *Office* platforms will be running *Office 200X*, our installation program will still be able to install a version of *Access runtime* that is compatible with our current version of HFACS -- in a manner that is nearly invisible to the user. Similarly, use of the MSDE ensures that a compatible data engine will be available. Finally, wherever possible, we tried to encapsulate program code outside of *Access* using completely object-oriented code. To this end, we created our own stand-alone connection component, completely isolated from *Access* specific connection operations. We provided our own FTP server, our own clipboard printing dynamic link library, our own password and security features, and our own initialization file for storing persistent data. Furthermore, for graphing operations, we used the *Visual Basic 6.0 Enterprise Edition's* MSChart Active-X library -- which, unlike the internal *Access 2000* charting objects is separately compiled and operates outside of *Access*. It is our intent that these measures provide the isolation from incompatibility associated with technology changes for at least five years.

**What new features should be implemented to make the information systems more user interactive and user friendly?**

Several changes were made to improve the usability of HFACS. The following list summarizes what we feel were the most dramatic:

- Support for dynamic screen resizing based upon the user's video resolution. By providing this support, the HFACS user interface can scale to different sizes for users with different size video monitors. This greatly improves the legibility of form data on all platforms.

- Elimination of separate menu options for query of data by a single or multiple fields. In the old version of HFACS, there were separate options for querying the database by single or multiple fields, this was due to the inability to effectively add "All" as a choice in queries. This limitation was overcome in the new version and we feel a great amount of redundancy in the user interface was removed.
- Similarly, the old version of HFACS provided separate menus for query by factor and for querying by summary of factors. We found this redundant and designed the factor summary so that individual text boxes on the summary form can be double-clicked to view detailed data pertaining to the mishaps
- Graph support in the old program consisted of only one type of 3D graph. This severely limited the usefulness of graphs as plots of large amounts of data were largely unreadable. The new version of HFACS has a much more robust graphing interface with support for 4 different style graphs, 2D & 3D representation, transposition of axes, stacking of data series, rotation of 3D graphs, and other improvements.

## B. RECOMMENDATIONS

As already discussed, we recommend further investigation of a middle-tier of software to support HFACS for use with Enterprise level *SQL Server* installations. Next, when *Visual Basic .NET* becomes widely available, we recommend investigation of improvements in the code base to port our existing code to its format. We believe this will significantly enhance the longevity of HFACS in terms of compatibility with newer versions of *Microsoft* products. Additionally, the following areas are good candidates for further research:

- Migration of our installation program from *Access 2000 SR-1 Runtime* to *Access XP Runtime*. This will eliminate the need for all *Office* service packs prior to installation of our program -- greatly improving ease of installation. *Access XP Runtime* is available only as part of the *Office XP Developer Edition* (retail ~\$799).
- Development of an Active X add-in to provide more robust report capabilities. A drawback of using *Access* reports is that they can only be previewed if a default printer has been specified in *Windows*. To circumvent this problem, an Active-X add-in should be developed to provide report preview functionality.
- The current version of HFACS uses database replacement as the means to update the official HFACS data. Research should be conducted into methods to update the existing data in the distributed instances of MSDE using replication, rather than database replacement. Replacement

HFACS.mdf and HFACAS\_log.ldf files can take up to an hour to download via FTP, whereas replication would take considerably less time as it only needs to add new and update the changed records -- instead of replacing them all.

- Investigation into the implementation of a key server to provide added security for add/edit operations should be conducted. The current User ID and Password files are stored in a hidden table in the Investigate.mdb file. As a result, every client has its own set of user IDs and Passwords. A key server would allow this data to be stored in a single location for all clients.
- Automated configuration of NTFS permissions. Installation of HFACS on *Windows 2000* systems using NTFS requires manual configuration of the program after installation in order to enable it for use by "domain users." Automated configuration is desirable, but will be considerably difficult to implement. It would require automated detection of domain names and automated configuration of user accounts with reference to security groups and file permissions.

### **C. SUMMARY**

Throughout this thesis, we have discussed many of the different alternatives considered in the development of the new HFACS client/server system. Techniques were described to provide sound documentation of our research, process logic, and implementation decisions. We believe that our solution provides the best mix of performance, scalability, and compatibility to meet the requirements of our sponsor. From this stage, HFACS is ready for independent usability study, fielding, and follow-on development cycles to add more functionality. We hope that the code that we worked so hard to develop will not be the first code that will need modification when new technologies become available -- we don't think it will. Nonetheless, in the event that it does, the meticulous software engineering described in this thesis should provide sound background for future changes, as well as, ample opportunity for code reuse.

THIS PAGE INTENTIONALLY LEFT BLANK

## APPENDIX A. CRC CARDS DEVELOPED FOR HFASC-ME

CRC cards developed for HFACS-ME are shown below.

### A. CONNECTION COMPONENT CRC CARDS

HFACS Connection	
Responsibilities:	Other Classes:
•Provide interface to all other classes and all program functionality	MSDE
	INIFile
	DiskUpdate
	FTPUpdate
	Logon

MSDE	
Responsibilities:	Other Classes:
•Start Server	Logon
•Stop Server	INIFile
•Attach Database File	
•Drop Database File	

Logon	
Responsibilities:	Other Classes:
•Log onto a specific instance of SQL server	MSDE
	INIFile

INIFile	
Responsibilities:	Other Classes:
•Read from an INI file	Logon
•Write to an INI file	

DiskUpdate	
Responsibilities:	Other Classes:
•Update database from a file on disk/network	Logon
	INIFile
	MSDE



FTPUpdate	
Responsibilities:	Other Classes:
•Update database from a	Logon
file downloaded via FTP	INIFile
	MSDE

FTP	
Responsibilities:	Other Classes:
•Perform internet FTP	FTPUpdate
functions	

## B. BUSINESS LOGIC COMPONENTS CRC CARDS

Aircraft	
Responsibilities:	Other Classes:
•Table manipulation for	Append
tblAircraft	Delete
	Find

Database Type	
Responsibilities:	Other Classes:
•Table manipulation for	Append
tblDatabaseType	Delete
	Find

Factors	
Responsibilities:	Other Classes:
•Table manipulation for	Append
tblFactors	Delete
	Find

Mishap Class	
Responsibilities:	Other Classes:
•Table manipulation for	Append
tblMishapClass	Delete
	Find

Mishap Factors	
Responsibilities:	Other Classes:
•Table manipulation for	Append
tblMishapFactors	Delete
	Find

Mishap Location	
Responsibilities:	Other Classes:
•Table manipulation for	Append
tblMishapLocation	Delete
	Find

Mishaps	
Responsibilities:	Other Classes:
•Table manipulation for	Append
tblMishaps	Delete
	Find

Mishap Type	
Responsibilities:	Other Classes:
•Table manipulation for	Append
tblMishapType	Delete
	Find

Organization	
Responsibilities:	Other Classes:
•Table manipulation for	Append
tblOrganization	Delete
	Find

Query Base Class	
Responsibilities:	Other Classes:
•Base Class for SQL	Append
query operations	Delete
	Find

Append	
Responsibilities:	Other Classes:
•Interface to base class for append query operations	Query Base Class

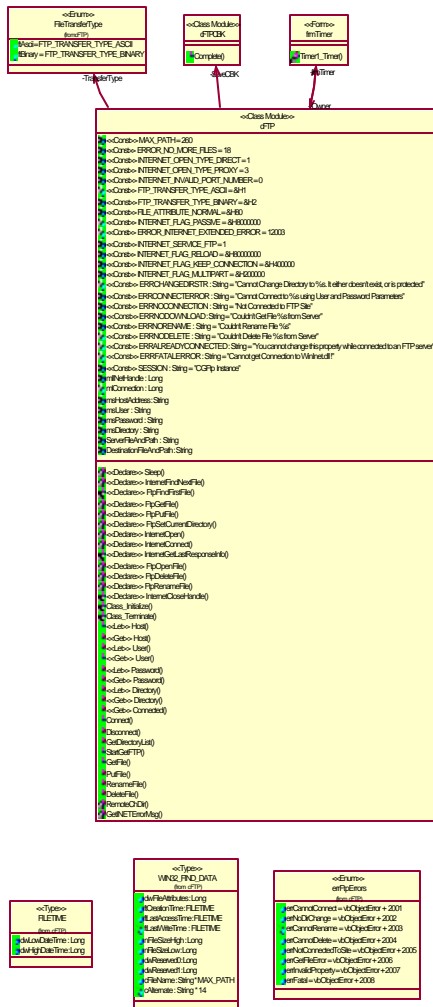
Delete	
Responsibilities:	Other Classes:
•Interface to base class for Delete query operations	Query Base Class

Find	
Responsibilities:	Other Classes:
•Interface to base class for Find query operations	Query Base Class

THIS PAGE INTENTIONALLY LEFT BLANK



## DIAGRAM





## APPENDIX C. DESCRIPTION OF CLASSES

### A. HFACS CONNECTION CLASS

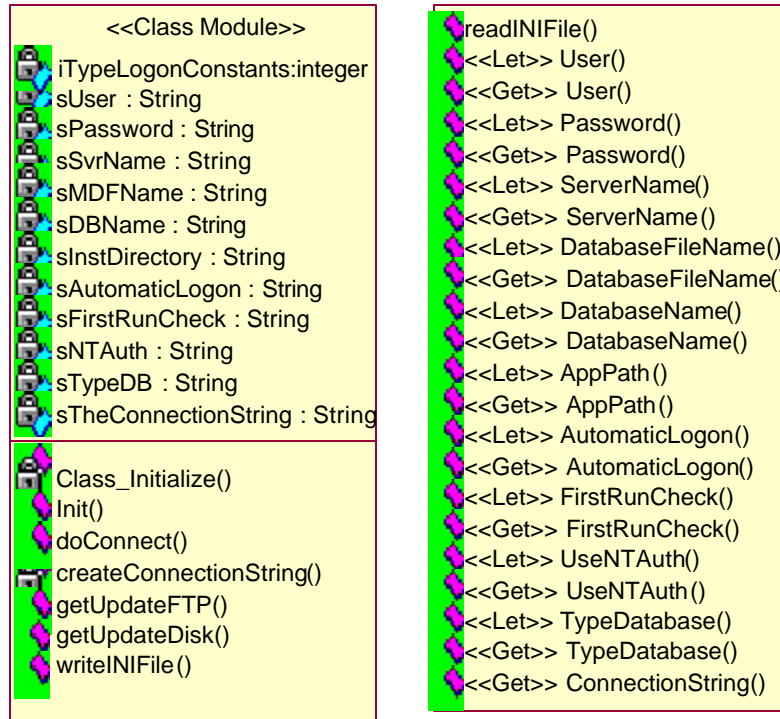


Figure C.1. Class Diagram for HFACS Connection.

#### 1. Class Description

This class is the controller class for the entire component. It is the only class with public members accessible from outside of the component. Nothing can be manipulated without creating an instance of this class and using its methods to indirectly utilize the functionality of the other classes.

#### 2. Data Member Description

**iTypeLogonConstants**--Enumerations for prompt/no-prompt functions in integer.

**sUser**--The user ID in string type.

**sPassword**--The user password in string type.

**sSvrName**--The name of the MSDE or SQL Server in string type.

**sMDFName**--The name of the .mdf file containing the database in string type.

**sDBName**--The name of the database in string type.

**sInstDirectory**--The application path in string type.

**sAutomaticLogon**--Toggle to log on with/without prompt in string type.

**sFirstRunCheck**--Toggle for determining if this is the first run after an update in string type.

**sNTAuth**--Toggle for determining if NT authentication should be used for logon attempts in string type.

**sTypeDB**--The type of DB this program will represent (mil, civilian, or both) in string type.

**sTheConnectionString**--Variable to hold the value of the current connectionstring in string type.

### 3. Method Description

**Class\_Initialize()** – Default no-argument constructor (initialize event).

**init()** --If an instance of a class is created using the psuedo-constructors from the Constructors.bas module, this function is called to pass initial values, thereby mimicking the behavior of a constructor with arguments. Passed in values are all required, but the Constructors.New\_HFACSConnection() function automatically sets passed-in values to global variable values if they are left blank.

**doConnect()**--This procedure will make a connection to a database server based on the value of iTypeLogonIn. If this parameter is left blank, the class determines the appropriate type of logon to perform. This function also detects if it is the first time HFACS has been run and displays the frmWelcome.frm as appropriate. After a successful logon, it sets the .ini value indicating a first run to "F."

**createConnectionString()**--This procedure updates the value of the global variable for the connection string that will be used for all ADO connections (hfacsmain.gTheConnectionString). It determines if the string should use NT authentication or regular SQL based on the global variable gStrNTauth.

**getUpdateFTP()**--This function creates an instance of the UpdateController class, providing access to FTP updates.

**getUpdateDisk()**--This function creates an instance of the UpdateController class, providing access to update from disk functionality.

**writeINIFile()**--This function creates an instance of the INIFileController class, providing methods to write to the HFACS.ini file.

**readINIFile()**--This function creates an instance of the INIFileController class, providing methods to read from the HFACS.ini file.

## B. ODBLOGON CLASS

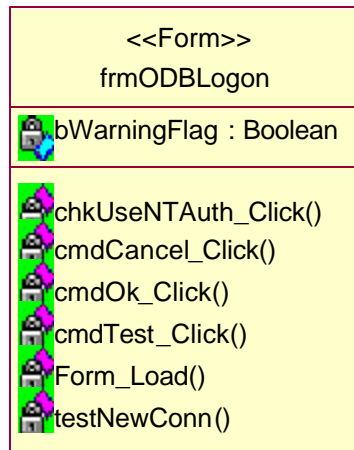


Figure C.2. Class Diagram for ODBLogon.

### 1. Class Description

This class is responsible for a prompted logon. We provide the capability to query a user for logon parameters and test their validity against a given instance of a SQL Server.

### 2. Data Member Description

**bWarningFlag**-- Warning flag indicating that the database needs to be installed on the local server in Boolean.

### 3. Method Description

**chkUseNTAuth\_Click()**--This sub updates form properties when the user clicks the "Use NT Authentication" check box. It "gray's out" the username and password text boxes and makes them unavailable for update.

**cmdCancel\_Click()**--This sub closes the form.

**cmdOk\_Click()**--This sub combines the functionality of testing the connection with the user supplied parameters and, if the parameters are valid, updating the pertinent global variables to enable other component class instances to function (e.g. to update the .ini file with new settings).

**cmdTest\_Click()**--This sub calls the testNewConn() function and returns an appropriate message to the user.

**Form\_Load()**--This sub sets the states of the form controls (visible/ not visible and enabled/ disabled) based upon current global variable settings.

**testNewConn()**--This sub tests the validity of the user specified connection values by attempting to start and connect to the server. Upon successful connection to the server specified, it verifies existence of the HFACS database on that server.

## C. UPDATECONTROLLER CLASS

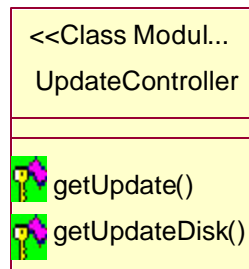


Figure C.3. Class Diagram for UpdateController Class.

### 1. Class Description

This class is the controller class for the cFTP class, the FTP form (frmFTPUpdate), and the common dialog control for reading an update from a disk.

### 2. Data Member Description

None.

### 3. Method Description

**getUpdate()**--This function initiates the FTP update session by creating an instance of frmFtpUpdate which actually performs the download and update.

**getUpdateDisk()**--This function displays the "Open" dialog box from the Microsoft Windows Common Controls 6.0 allowing the user to identify a path on a disk/network share where the HFACS.mdf/\_log.ldf update files reside. It then copies the files to the application path on the local machine and instantiates an instance of frmDiskUpdate to install them.

### D. DISK UPDATE CLASS

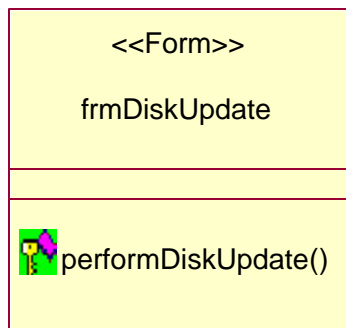


Figure C.4. Class Diagram for Disk Update Class.

### 1. Class Description

This class is responsible for performing an update of the HFACS database from a disk/network share.

### 2. Data Member Description

None

### 3. Method Description

**performDiskUpdate()**--This function performs the actual update, updating the form as it progresses.

## E. FTPUPDATE CLASS

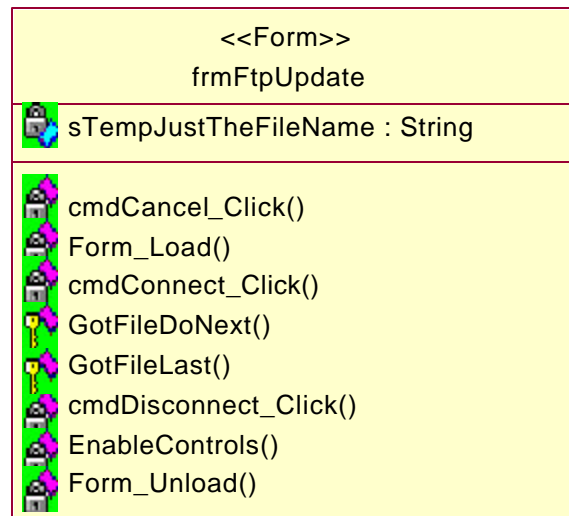


Figure C.5. Class Diagram for FTPUpdate Class.

### 1. Class Description

This class is responsible for performing an update of the HFACS database via FTP. This class uses the FTPServer.exe server and the CallbackCls.cls to receive status messages from the HFACS FTP server. The FTP server (HFACSFTP.exe) provides the functions needed to get FTP updates. These functions and their associated classes were removed from this component and compiled separately in order to work around the inability of Visual Basic to provide support for free threading. By placing the FTP functionality in a separately compiled executable, it can run in it's own process, which allows screen updates during long FTP downloads.

### 2. Data Member Description

**STempJustTheFileName**-- A temp string variable to simplify string manipulation when determining paths on the FTP server and for download locations.

### 3. Method Description

**cmdCancel\_Click()**--This sub closes the form.

**Form\_Load()**--This sub resets flags when the form is opened.

**cmdConnect\_Click()**--This sub verifies that the FTP is being performed on a local server and initiates the FTP connection by instantiating an FTP server object. It then downloads the first new database file (HFACS.mdf) to the application path. When

download of the first file is complete, the CallbackCls interface is notified by the FTP server, which in turn executes the download of the next file via the GotFileDoNext() sub.

**GotFileDoNext()**--This sub downloads the second new database file (HFACS\_log.ldf) to the application path. When download of the file is complete, the CallbackCls interface is notified by the FTP server, which in turn executes the installation of the 2 files via the GotFileLast() sub.

**GotFileLast()**--This sub performs the actual update, updating the form to show status as it progresses.

**cmdDisconnect\_Click()**--This sub performs disconnect from the FTP server when it is enabled. It is not enabled except during development.

**EnableControls()**--This sub performs dynamically enables/disables buttons on the form based upon the connection state of the FTP server.

**Form\_Unload()**--This sub performs cleanup operations, ensuring all objects are destroyed when the form is closed.

## F. MSDE CLASS

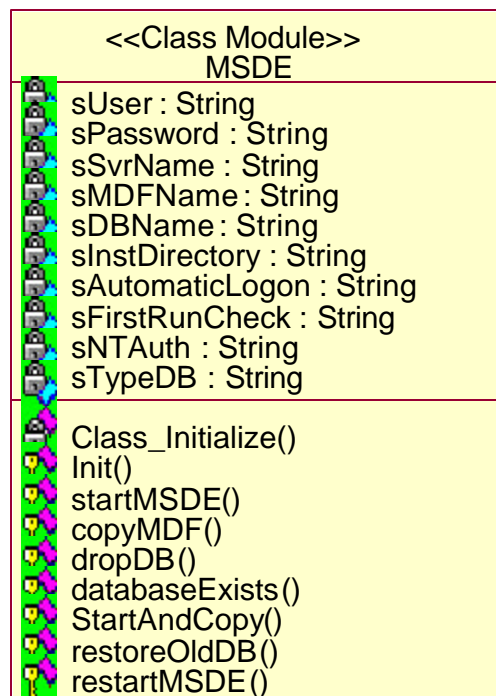


Figure C.6. Class Diagram for MSDE Class.

### 1. Class Description

This class is responsible for starting the MSDE or SQL server, ensuring that the HFACS database is installed, and managing database updates.

### 2. Data Member Description

**sUser**--The user ID in string type.

**sPassword**--The user password in string type.

**sSvrName**--The name of the MSDE or SQL Server in string type.

**sMDFName**--The name of the .mdf file containing the database in string type.

**sDBName**--The name of the database in string type.

**sInstDirectory**--The application path in string type.

**sAutomaticLogon**--Toggle to log on with/without prompt in string type.

**sFirstRunCheck**--Toggle for determining if this is the first run after an update in string type.

**sNTAuth**--Toggle for determining if NT authentication should be used for logon attempts in string type.

**sTypeDB**--The type of DB this program will represent (mil, civilian, or both) in string type.

### 3. Method Description

**Class\_Initialize()**—Default no argument constructor (initialize event).

**Init()**--If an instance of a class is created using the psuedo-constructors from the Constructors.bas module, this function is called to pass initial values, thereby mimicking the behavior of a constructor with arguments. Passed in values are all required, but the Constructors.New\_MSDE() function automatically sets passed-in values to global variable values if they are left blank.

**startMSDE()**--This procedure will start an instance SQL Server and create a connection to it, thereby verifying that the specified server exists and that it is started. If the server is already running, the error trap will exit the procedure and leave the server running. A bug in SQL Server 2000 prevents SQLDMO from starting a remote server so



this code also detects the error and switches to an ADO type connection to verify that the HFACS database is present on the remote machine. In the case of the ADO connection, a copy the database either exists or doesn't exist on the remote server. If the ADO connection fails, a global flag is set so that all classes in the component know not to try to copy an instance of the database to the remote server, which would generate another error.

**copyMDF()**--This procedure will check for the database on a local Server. If the database does not exist, it will then copy and install the HFACS database from the application path to the Server data directory making a backup copy of the old database in case an error occurs and a restore is needed. The last two copies of the database are kept in the server data directory in an attempt to prevent data loss.

**dropDB()**--This procedure will check for the database on the Server. If the database exists it will then permanently drop it. A normal drop specifies the bKillDBFiles parameter as False, so a backup of the database is created before dropping it. Passing a value of true for this parameter drops the database with no backup.

**databaseExists()**--This procedure will connect to a SQL server that is already running and determine if a database exists.

**StartAndCopy()**--This procedure combines the functionality of the startMSDE() and copyMDF() functions with the added ability to determine if a copy is needed based upon the results of the startMSDE() call. For example, if a remote connection is attempted and succeeds, startMSDE() will return True, but no copy will be necessary. In addition, this function detects if a copy failed and will attempt to repair the database by offering an option to restore an old copy of the database. This is useful when called from a failed FTP update attempt.

**restoreOldDB()**--This function is called when a copy operation fails and there is no HFACS database file attached to the local server. Once called, this function prompts the user to restore the old database. If the user opts to restore the database, a restore is first attempted using the current logon information. If this attempt fails, a second attempt is made as a "last-ditch" effort using the "sa" logon and no password. If both attempts fail, the database will not be installed on the local server and the HFACS program will

not function. System Administrator assistance will be required to attach a copy of the database.

**restartMSDE()**--Before an .mdf database file can be dropped and a new file attached, all users must be logged off. This function stops and restarts the server effectively ensuring all users are logged off and that the server services are refreshed. This function can only be used in conjunction with an update operation (either disk or FTP) as it also copies the file from the download/temp copy directory (which is the application path) to the server data directory. This copy can only be performed when the server is stopped.

## G. CALLBACK CLASS

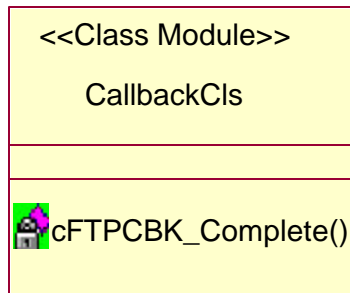


Figure C.7. Class Diagram for Callback Class.

### 1. Class Description

This class implements the cFTPCBK callback interface of the HFACS FTP server. The methods of this class provide the means for the HFACS server to notify (or callback) class instances from this component which utilize the FTP server functionality. Basically, the members of this class provide a communication channel. The FTP server (HFACSFTP.exe) provides the functions needed to get FTP updates. These functions and their associated classes were removed from this component and compiled separately in order to work around the inability of Visual Basic to provide support for free threading. By placing the FTP functionality in a separately compiled executable, it can run in it's own process, which allows screen updates during long FTP downloads.

### 2. Data Member Description

None

### 3. Method Description

**cFTPCBK\_Complete()**--An FTP update of the HFACs database requires the download of 2 files (HFACS.mdf & HFACS\_log.ldf). This function accepts messages from the FTP server and notifies the frmFtpUpdate of progress. Specifically, of errors in download and of successful download. If the first file is downloaded successfully (ErrCode = True And gIntCounter = 1), then this function notifies the frmFtpUpdate to begin the next download. After successfully downloading both files, this function closes the frmFtpUpdate form.

### H. INIFILE CLASS

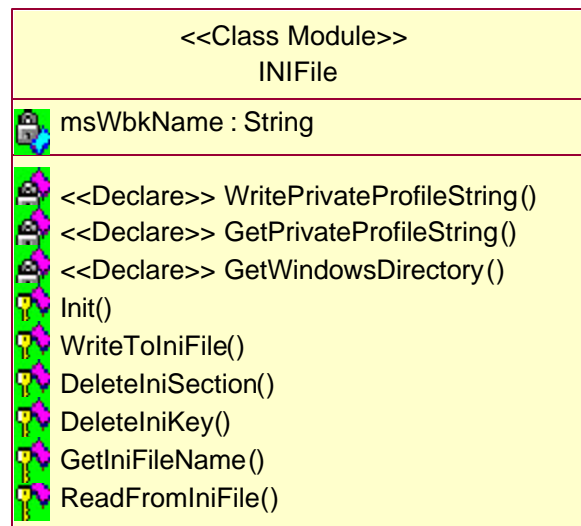


Figure C.8. INIFile Class Diagram.

### 1. Class Description

This class creates .ini File objects used to create, delete, set, and get values in a standard format Microsoft .ini file. It uses calls to the Windows API for efficiency.

### 2. Data Member Description

**msWbkName**--The name of the ini file to read in string type.

### 3. Method Description

**Init()**--If an instance of a class is created using the psuedo-constructors from the Constructors.bas module, this function is called to pass initial values, thereby mimicking the behavior of a constructor with arguments. Passed in values are all required, but the

Constructors.New\_INIFile() function automatically sets passed-in values to global variable values if they are left blank.

**WriteToIniFile()**--Write a section, key, and value to an .ini file.

**DeleteIniSection()**--Delete a section and all of its keys from an .ini file.

**DeleteIniKey()**--Delete a key and its value from an .ini file.

**GetIniFileName()**--Return name for .ini file. Name includes name of workbook file and ".ini" extention.

**ReadFromIniFile()**--Read a value from an .ini file, given the file name, section, key, and default value to return if key is not found.

## I. HFACSMAN CLASS

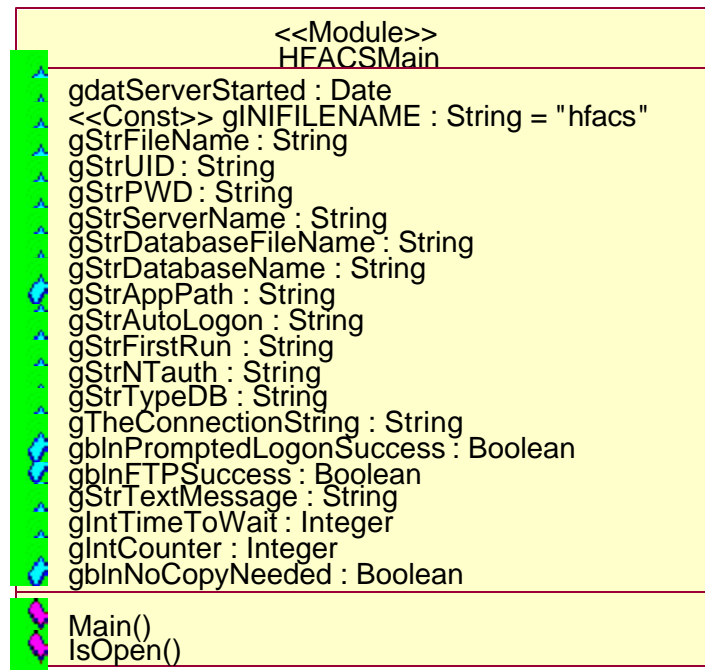


Figure C.9. HFACSMAN Class Diagram.

### 1. Class Description

This module is accessible to all classes and forms in the project. It contains declarations for all global variables used to pass values between forms and instances of classes.

## 2. Data Member Description

**gdatServerStarted**--This variable is used by HFACSMian.Main() for initializing the entire component. It is required for all compiled DLLs, but not used for anything else. It is a date type.

**gINIFILENAME**--Constant variable to hold the name of the .ini file. It is a string type and its value is “hfacs”. This is a global variable.

**oINIFile**-- Reusable object variables. These variables are used over and over by classes and forms. They are created and destroyed within the same function whenever possible. It is an instance of INIFile and it has global scope.

**oINIFileController**--Reusable object variable for the INI file control class. This is a global variable.

**oHFACSConnection**--Reusable object variable for the HFACSConnection class. This is a global variable.

**oMSDE**--Reusable object variable for the MSDE Class. This is a global variable.

**oUpdateController**--Reusable object variable for the UpdateController Class. This is a global variable.

**gStrFileName**--Global variable to hold the path to the Windows system directory. This is a string type.

**gStrUID**--Global variable representing the user ID in string type.

**gStrPWD**--Global variable representing the user password in string type.

**gStrServerName**--Global variable representing the name of the MSDE or SQL Server as string type.

**gStrDatabaseFileName**--Global variable representing the name of the mdf file as string type.

**gStrDatabaseName**--Global variable representing the name of the database as string type.

**gStrAppPath**--Global variable representing the application path as string type.

**gStrAutoLogon**--Global variable to toggle to logon without prompt as string type.

**gStrFirstRun**--Global variable representing the toggle for determining the first time the program has been run as string type.

**gStrNTauth**--Toggle for determining if NT authentication should be used for logon attempts as string type.

**gStrTypeDB**--The type of DB this program will represent (mil, civ, or both) as string type.

**gTheConnectionString**--Global variable to hold the value of the current connectionstring as string type.

**gSQLServerPath**--Global variable to hold the value of the SQL Server subdirectory as string type.

**gblnPromptedLogonSuccess**--Boolean that indicates a success/failure of a prompted logon.

**gblnFTPSuccess**--Boolean that indicates a success/failure of an FTP update attempt.

**gStrTextMessage**--A string type that holds a message for label on frmWait. Allows you to change the message from any location in this component.

**gIntTimeToWait**--An integer variable that represents the amount of time for frmWait to count. Allows you to set the number of seconds for frmWait to actually wait.

**gIntCounter**--Reusable integer variable for counters throughout the component.

**gblnNoCopyNeeded**--Boolean for indicating no copy is necessary. This is required when making a connection to a remote host because the SQL Server 2000 version of SQLDMO won't connect to a remote host. To work around this, an ADO connection is attempted. If an ADO connection succeeds, then the database exists on the server being connected to, so no copy is needed . . . and this boolean is set.

*a) Method Description.*

**Main()**--This code is executed when the component starts, in response to the first object request. It is the "Main" procedure responsible for initializing the entire component and is required for all compiled DLLs.

**IsOpen()**--Determines if a form is open or not. Useful for determining when screen refreshes are needed.

## J. INIFILECONTROLLER CLASS

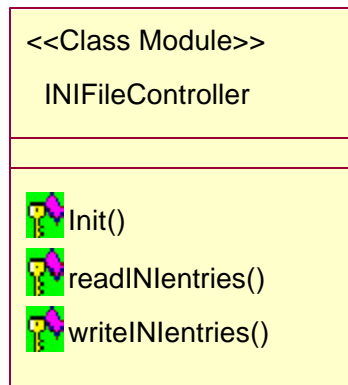


Figure C.10. INIFileController Class Diagram.

### 1. Class Description

This class creates instances of **INIFile.cls** used to create, delete, set, and get values in a standard format Microsoft **.ini** file.

### 2. Data Member Description

None

### 3. Method Description

**Init()**--If an instance of a class is created using the psuedo-constructors from the **Constructors.bas** module, this function is called to pass initial values, thereby mimicking the behavior of a constructor with arguments. Passed in values are all required, but the **Constructors.New\_INIFileController()** function automatically sets passed-in values to global variable values if they are left blank.

**readINIentries()**--This function creates an instance of the **INIFile** class and reads values from the **HFACS.ini** file.

**writeINIentries()**--This function creates an instance of the **INIFile** class and writes values to the **HFACS.ini** file.

## K. WAIT CLASS

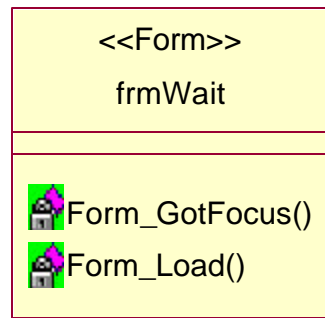


Figure C.11. Wait Class Diagram.

### 1. Class Description

This class is responsible for showing a status bar capable of pausing the number of seconds specified by HFACSMMain.gIntTimeToWait and displaying the message contained in HFACSMMain.gStrTextMessage.

#### 1. Data Member Description

None

#### 2. Method Description

**Form\_GotFocus()**--This sub reads the values contained in the global variables to determine how long to show itself and what message to display.

**Form\_Load()**--This sub reads the values contained in the global variables to determine the message to display on the form.

## L. WELCOME CLASS

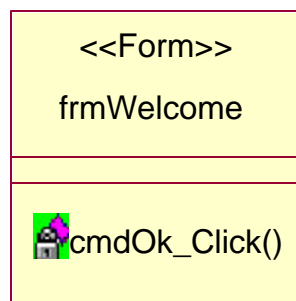


Figure C.12. Welcome Class Diagram.

### 1. Class Description

This class is responsible for displaying an all text welcome message when it is called.



## 2. Data Member Description

None

## 3. Method Description

**cmdOk\_Click()**--This function unloads this form once the user clicks the OK button.

## M. CONSTRUCTORS CLASS

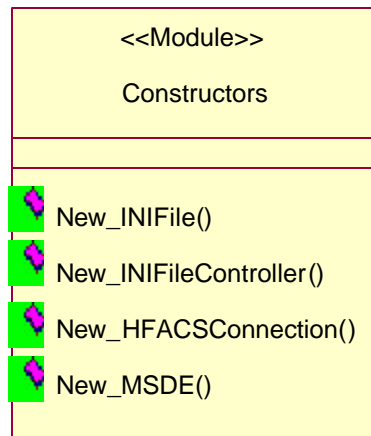


Figure C.13. Constructors Class Diagram.

## 1. Class Description

This module defines functions that pair creation of new object instances using the reusable global objects defined in HFACSMMain class with a call to an Init() function of the associated class. In this manner, these functions can act as psuedo-constructors that are capable of passing arguments -- a feature not available in Visual Basic 6.0.

## 2. Data Member Description

None

## 3. Method Description

**New\_INIFile()**--This function acts as a psuedo-constructor. It creates a new INIFile object and calls the INIFile.Init() function, passing desired parameters to ensure a consistent state.

**New\_INIFileController()**--This function acts as a psuedo-constructor. It creates a new INIFileController object and calls the INIFileController.Init() function, passing desired parameters to ensure a consistent state.

**New\_HFACSCConnection()**--This function acts as a psuedo-constructor. It creates a new HFACSCConnection object and calls the HFACSCConnection.init() function, passing desired parameters to ensure a consistent state.

**New\_MSDE()**--This function acts as a psuedo-constructor. It creates a new MSDE object and calls the MSDE.Init() function, passing desired parameters to ensure a consistent state.

## N. ERRORLOG CLASS

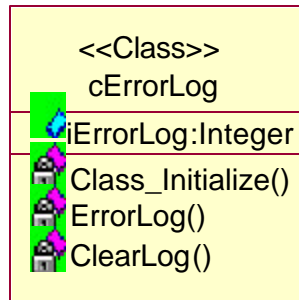


Figure C.14. ErrorLog Class Diagram.

### 1. Class Description

This writes status and error messages to the App.path connectionErrors.log file.

### 2. Data Member Description

**iErrorLog**--Integer value for each entry

### 3. Method Description

**Class\_Initialize()**--Default no-argument constructor for (initialize event).

**ErrorLog()**--Open the a file called ConnectionErrLog.log in the application path and write error entries to it.

**ClearLog()**--Clears the ConnectionErrLog.log.

## O. FTPCBK CLASS

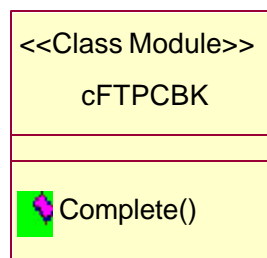


Figure C.15. FTPCBK Class Diagram.

### 1. Class Description

The method of this class provide the means for the HFACS server to notify (or callback) class instances from this component which utilize the FTP server functionality. The FTP server (HFACSFTP.exe) provides the functions needed to get FTP updates. These functions and their associated classes were compiled separately in order to work around the inability of Visual Basic to provide support for free threading. By placing the FTP functionality in a separately compiled executable, it can run in it's own process, which allows screen updates during long FTP downloads.

### 2. Data Member Description

None

### 3. Method Description

**Complete()**--An FTP update of the HFACs database requires the download of 2 files (HFACS.mdf & HFACS\_log.ldf). This function sends messages from the FTP server and notifies the Callback class of the progress. Specifically, of errors in download and of successful download.

## P. TIMER CLASS

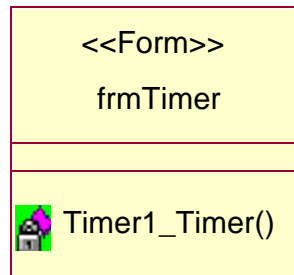


Figure C.16. Timer Class Diagram.

### 1. Class Description

This class disables the timer as the FTP class initiates the change in

### 2. Data Member Description

None

### 3. Method Description

**Timer1\_Timer()**--This procedure is executed only once per each invocation and disables the timer.

## Q. FTP CLASS

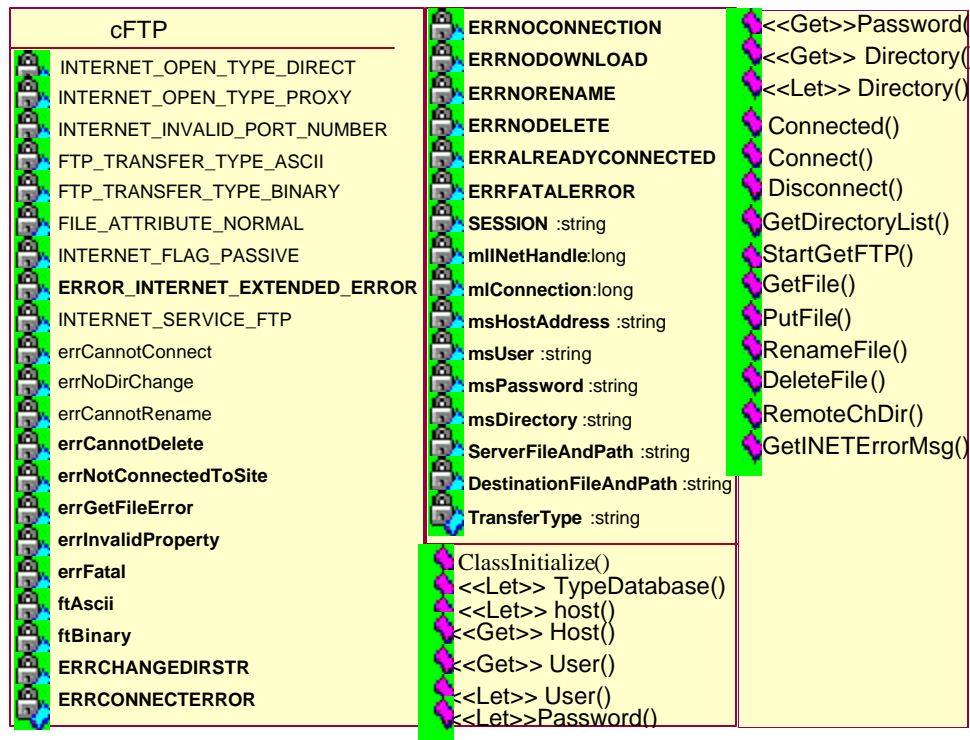


Figure C.17. FTP Class Diagram.

### 1. Class Description

This class wraps the functionality of the Win32 WinInet.DLL. It could easily be expanded to provide HTTP/Gopher and other internet standard file protocols.

### 2. Data Member Description

**INTERNET\_OPEN\_TYPE\_DIRECT**--Constant variable for registry access settings.

**INTERNET\_OPEN\_TYPE\_PROXY**--Constant variable for registry access settings.

**INTERNET\_INVALID\_PORT\_NUMBER**-- Constant variable for registry access settings.

**FTP\_TRANSFER\_TYPE\_ASCII**-- Constant variable for registry access settings.

**FTP\_TRANSFER\_TYPE\_BINARY**-- Constant variable for registry access settings.

**FILE\_ATTRIBUTE\_NORMAL**-- Constant variable for registry access settings.

**INTERNET\_FLAG\_PASSIVE**-- Constant variable for registry access settings.

**ERROR\_INTERNET\_EXTENDED\_ERROR**--Constant variable for error message.

**INTERNET\_SERVICE\_FTP**--Constant variable for the type of service to access.

**errCannotConnect**--Variable representing a type of FTP error.

**errNoDirChange** -- Variable representing a type of FTP error.

**errCannotRename** -- Variable representing a type of FTP error.

**errCannotDelete**-- Variable representing a type of FTP error.

**errNotConnectedToSite**-- Variable representing a type of FTP error.

**errGetFileError**-- Variable representing a type of FTP error.

**errInvalidProperty**-- Variable representing a type of FTP error.

**errFatal**-- Variable representing a type of FTP error.

**ftAscii**--File transfer type (ASCII)

**ftBinary**-- File transfer type (Binary)

**ERRCHANGEDIRSTR**--Constant variable of string type with an error message.

**ERRCONNECTERROR**-- Constant variable of string type with an error message.

**ERRNOCONNECTION**-- Constant variable of string type with an error message.

**ERRNODOWNLOAD**-- Constant variable of string type with an error message.

**ERRNORENAME**-- Constant variable of string type with an error message.

**ERRNODELETE**-- Constant variable of string type with an error message.

**ERRALREADYCONNECTED**-- Constant variable of string type with an error message.

**ERRFATALERROR**-- Constant variable of string type with an error message.

**SESSION**-- Constant string variable that identifies the session to Windows.

**mlInetHandle**--Long variable that identifies the INet handle.

**mlConnection**--Long variable that identifies the connection handle.

**msHostAddress**--String variable for standard FTP properties for this class.

**msUser**-- String variable for standard FTP properties for this class.

**msPassword**-- String variable for standard FTP properties for this class.

**msDirectory**-- String variable for standard FTP properties for this class.

**ServerFileAndPath**--String variable that holds the server file path.

**DestinationFileAndPath**--String variable that holds the destination file path for the downloaded files.

**TransferType**--String variable that indicates the type of transfer(ftp or disk).

### 3. Method Description

**Class\_Initialize()**--Create Internet session handle.

**Class\_Terminate()**--Kill off any connection and API handle.

**<let> Host()**-- Set the Host Name - only if not connected.

**<get> Host()**--Get the host name.

**<let> User()**--Set the user - if not connected.

**<get> User()**--Get the user name.

**<let> Password()**--Set the password - only if not connected.

**<get> Password()**--Get the user password.

**<let> Directory()**--Set the directory- only if not connected.

**<get> Directory()**--Get the directory.

**Connected()**--Indicates whether the system is connected to a server. It returns a boolean.

**Connect()**--This function connects to the FTP server. It will raise an error if the system is already connected.

**Disconnect()**--This function disconnect from the FTP server only if the system is currently connected.

**GetDirectoryList()**--Returns a Disconnected record set for the directory and filter string.

**StartGetFTP()**--This function establishes the variables to start an FTP session.

**GetFile()**--Get the specified file to the desired location using the specified file transfer type. This code is executed when the timer fires for the first time. It unloads the form and destroys it completely.

**PutFile()**--This function copies the files to the desired path specified in the parameter list of this function.

**RenameFile()**--This function renames the existing files for backup purpose. This function maintains two backup files.

**DeleteFile()**--This function deletes the oldest backup as the files are copied.

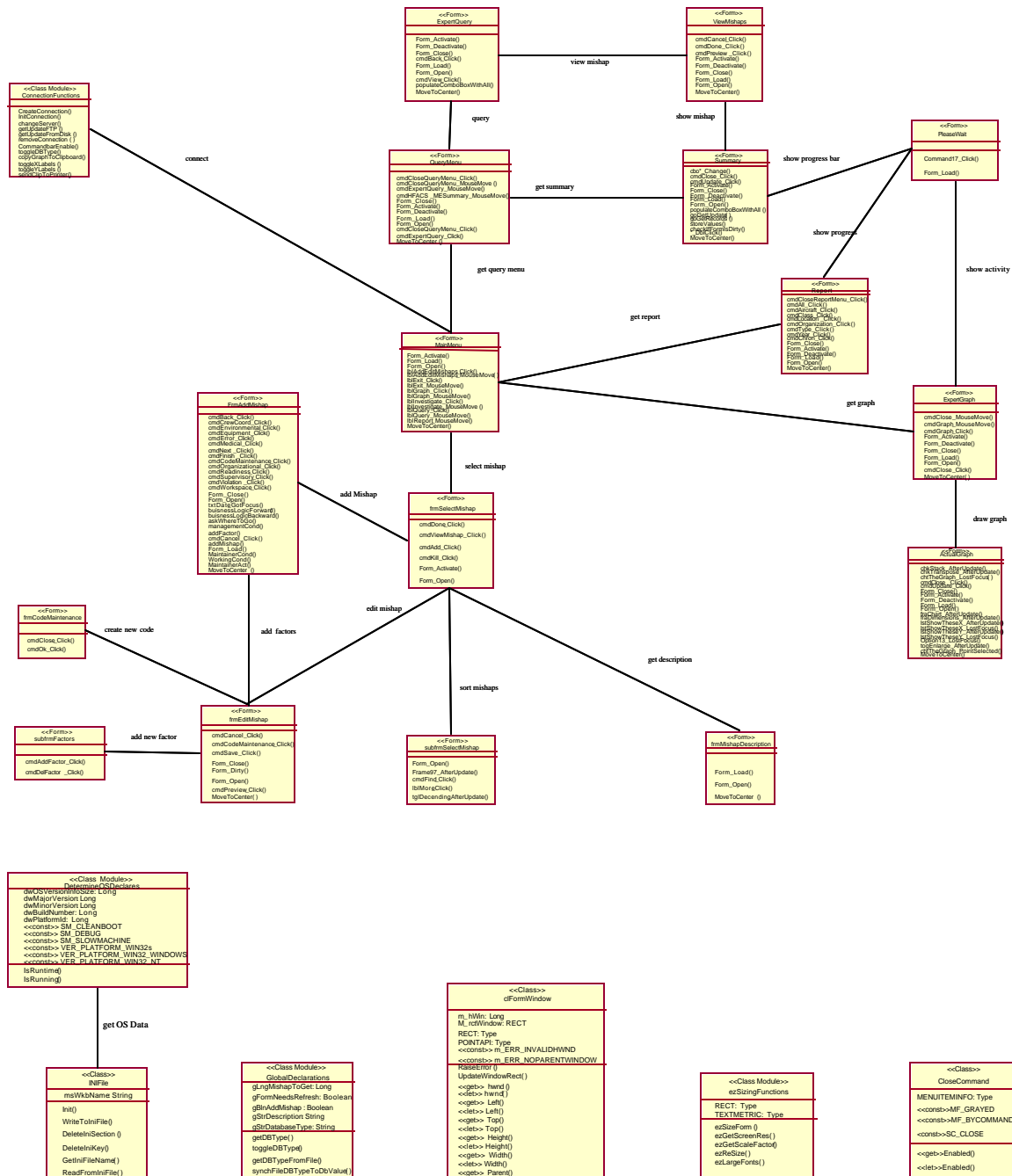
**RemoteChDir()**--This function changes the directory remotely.

**GetINETErrorMsg()**--Returns an error message indicating type of error that had occurred.

THIS PAGE INTENTIONALLY LEFT BLANK



## APPENDIX D.



THIS PAGE INTENTIONALLY LEFT BLANK

## APPENDIX E. DESCRIPTION OF BUSINESS LOGIC CLASSES

### A. INIFILE CLASS

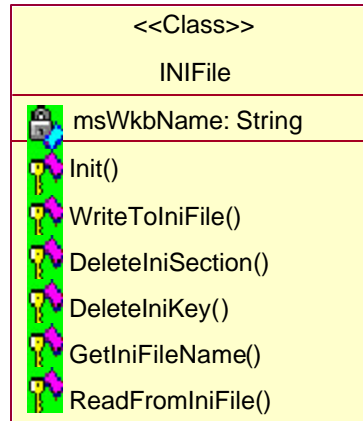


Figure E.1. Class Diagram for INIFile Class.

#### 1. Class Description

This class creates .ini File objects used to create, delete, set, and get values in a standard format Microsoft .ini file. It uses calls to the Windows API for efficiency.

#### 2. Data Member Description

**msWkbName**--The name of the ini file to read as a string type.

#### 3. Method Description

**Init()** -- If an instance of a class is created using the psuedo-constructors from the Constructors.bas module, this function is called to pass initial values, thereby mimicking the behavior of a constructor with arguments. Passed in values are all required, but the Constructors.New\_INIFile() function automatically sets passed-in values to global variable values if they are left blank..

**WriteToINIFile()** --Write a section, key, and value to an .ini file.

**DeleteINISection()**--Delete a section and all of its keys from an .ini file.

**DeleteINIKey()**--Delete a key and its value from an .ini file.

**GetIniFileName()**--Return name for .ini file. Name includes name of workbook file and ".ini".

**ReadFromIniFile()**--Read a value from an .ini file, given the file name, section, key, and default value to return if key is not found.

## B. GLOBALDECLARATIONS CLASS

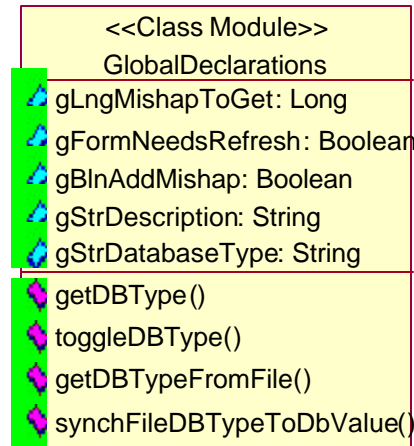


Figure E.2. Class Diagram for GlobalDeclaration Class.

### 1. Class Description

Contains all definitions for application global variables. Most of these are needed due to the inability of VBA to pass parameters as part of a constructor.

### 2. Data Member Description

**gLongMishapToGet** -- The ID of the mishap to read as long type.

**gFormNeedsRefresh** -- Indicates that the form needs to be refreshed as Boolean type.

**gBlnAddMishap** -- Indicates .a mishap has been added as Boolean type.

**gStrDescription** -- Holds the value if the description detail is too long to be held in initial text field as string type.

**gStrDatabaseTyp** -- Holds the value of the database type (civilian or military) as string type.

### 3. Method Description

**getDBType()**--Determines the type of database (military or civilian) based on the SQL severer tblDatabaseType settings.

**toggleDBType()**--Toggles the current investigation module DB type.

**getDBTypeFromFile()**--Determines the type of database (military or civilian) based on the HFACS.ini file settings.

**snchFileDBTypeToDbValue()**--Ensures that this program opens in the same mode (civilian or military) as the HFACS instance that launched it.

### C. DETERMINEOSDECLARES CLASS

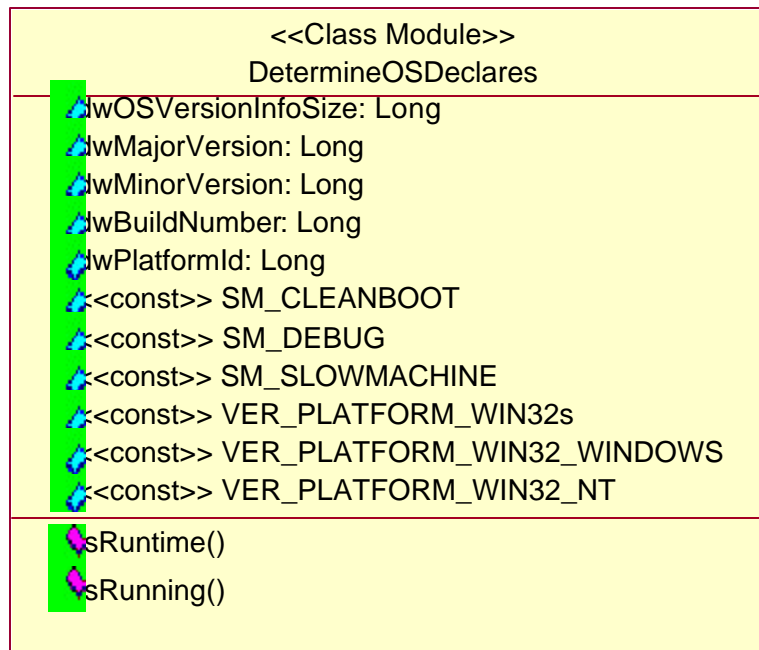


Figure E.3. Class Diagram for DetermineOSDeclares Class.

#### 1. Class Description

Contains various functions for determining system properties like O/S type and version of Access that is running.

#### 2. Data Member Description

**dwOSVersionInfoSize** -- Holds the operating version information that pertains to size as long type.

**dwMajorVersion** -- Holds the operating version information as long type.

**dwMinorVersion** -- Holds the operating version information as long type.

**dwBuildNumber** -- Holds the operating version information that pertains to the build as long type.

**dwPlatformId** -- Holds the operating version information as long type.

**SM\_CLEANBOOT** -- Constant variable that holds the value 67.

**SM\_DEBUG** -- Constant variable that holds the value 22.

**SM\_SLOWMACHINE** -- Constant variable that holds the value 73.

**VER\_PLATFORM\_WIN32s** -- Constant variable that holds the value 0.

**VER\_PLATFORM\_WIN32\_WINDOWS** -- Constant variable that holds the value 1.

**VER\_PLATFORM\_WIN32\_NT** -- Constant variable that holds the value 2.

### **3. Method Description**

**IsRuntime()**--Determines if Access runtime is being used to run the application. Access runtime has no support for reports.

**IsRunning()**--To prevent a second instance from loading if a user mistakenly attempts to launch it twice. This code is called from the autoexec macro to test whether the app is already running and terminate the launch if a copy of it is already open.

## D. FORMWINDOW CLASS

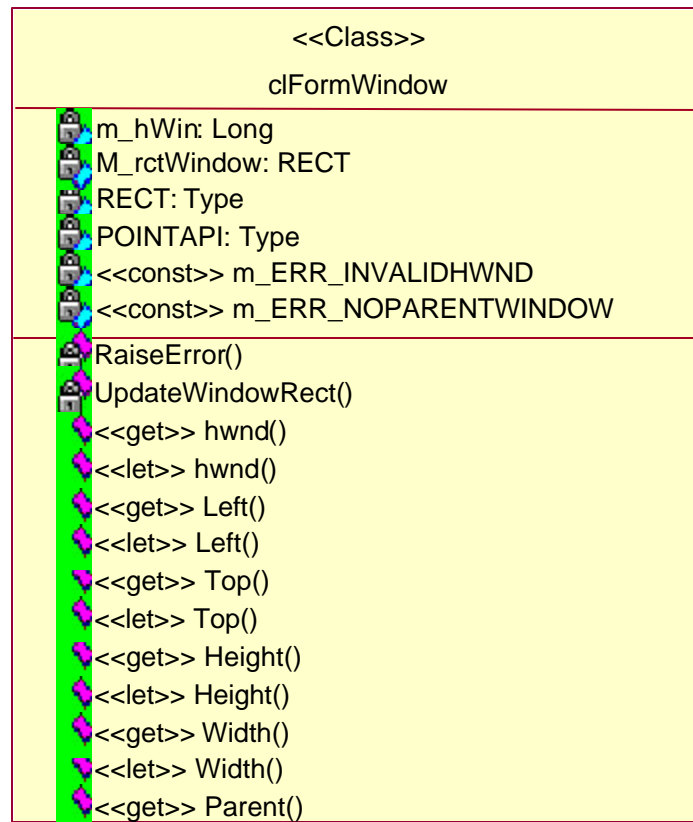


Figure E.4. Class Diagram for FormWindow Class.

### 1. Class Description

Moves and resizes a window in the coordinate system of its parent window.

### 2. Data Member Description

**m\_hWin** -- Handle of the window as long.

**m\_RctWindow** -- Rectangle describing the sides of the last polled location of the window as a rectangle type.

**RECT** -- RECT structure used for API calls.

**POINTAPI**—POINTAPI structure used for API calls.

**m\_ERR\_INVALIDHWND** -- Private error constants for use with `RaiseError` procedure. Holds value of 1.

**m\_ERR\_NOPARENTWINDOW** -- Private error constants for use with `RaiseError` procedure. Holds value of 2.

### 3. Method Description

**RaiseError()**--Raises a user-defined error to the calling procedure.

**UpdateWindowRect ()**--Places the current window rectangle position (in pixels, in coordinate system of parent window) in `m_rectWindow`.

**<<get>> hwnd()** -- Returns the value the user has specified for the window's handle.

**<<let>> hwnd()** -- Sets the window to use by specifying its handle. Only accepts valid window handles.

**<<get>> Left()** -- Returns the current position (in pixels) of the left edge of the window in the coordinate system of its parent window.

**<<let>> Left()** -- Moves the window such that its left edge falls at the position indicated (measured in pixels, in the coordinate system of its parent window).

**<<get>> Top()** -- Returns the current position (in pixels) of the top edge of the window in the coordinate system of its parent window.

**<<let>> Top()** -- Moves the window such that its top edge falls at the position indicated (measured in pixels, in the coordinate system of its parent window).

**<<get>> Width()** -- Returns the current width (in pixels) of the window.

**<<let>> Width()** -- Changes the width of the window to the value provided (in pixels).

**<<get>> Height()** -- Returns the current height (in pixels) of the window.

**<<let>> Height()** -- Changes the height of the window to the value provided (in pixels).

**<<get>> Parent()** -- Returns the parent window as a `clFormWindow` object. For forms, this should be the Access MDI window.



## E. SIZING FUNCTION CLASS

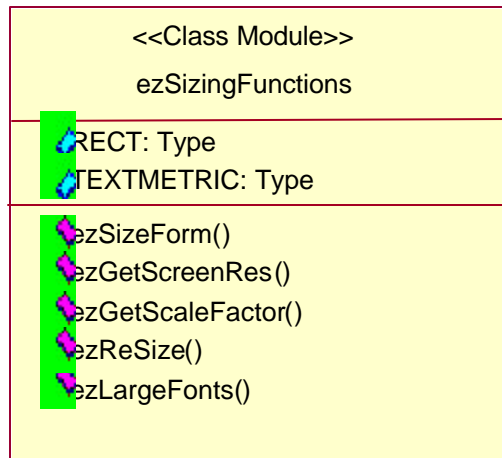


Figure E.5. Class Diagram for Sizing Function Class.

### 1. Class Description

Contains various functions for dynamically resizing the forms in the application based on the user's screen resolution. Created by EZ Sizing Functions, Copyright (C) 2000 Database Creations, Inc. Revision 6/14/00 based on 8/25/99 code with revisions.

### 2. Data Member Description

**RECT** -- RECT structure used for API calls.

**TEXTMETRIC**—TEXTMETRIC structure used for API calls.

### 3. Method Description

**ezSizeForm ()**--This subroutine will resize the form specified by parameter `xForm` by the factor of `ScaleFactor`. If scale factor is 0 or negative, automatic scaling will occur based on the following table.

Value	Forms originally designed for
0	640 x 480
-1	800 x 600
-2	1024 x 768
-3	1280 x 1024
-4	1600 x 1200
-5	1152 x 864 OR 1152 x 870

Table E.1. EzSizeForm Values.

**ezGetScreenRes()**--This function returns the windows screen size.

**ezGetScaleFactor ()**--Returns a scale factor for resizing based on the passed parameter S which should represent the screen size a form was designed for the scale factor returned is based on the current screen resolution.

**ezReSize ()**--This subroutine will resize the form based on its current dimensions.

**ezLargeFonts ()**--This function returns a true if large fonts are being used.

## F. SELECT MISHAP CLASS

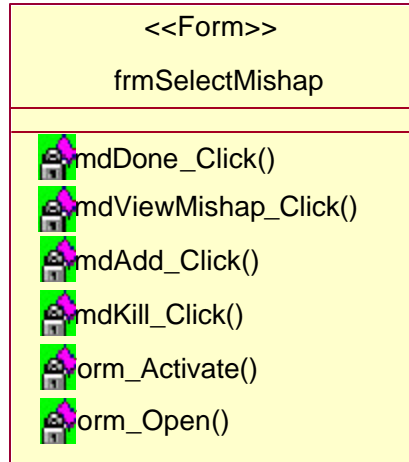


Figure E.6. Class Diagram for Select Mishap Class.

### 1. Class Description

This class displays all the Mishaps in the database and allows the user to sort them by various fields in order to select a mishap to view or edit. It has buttons that allow initiation of a new Mishap or deletion of an existing mishap.

## 2. Data Member Description

None.

## 3. Method Description

**cmdDone\_Click()** -- Closes the form.

**cmdViewMishap\_Click()**--Opens the mishap selected in the subform.

**cmdAdd\_Click()**--Opens the add mishap wizard.

**cmdKill\_Click()**--Deletes the mishap selected in the subform.

**Form\_Activate()**--Update the menu bar and see if the subform needs to be refreshed.

**Form\_Load()**--Dynamically resizes the form to the users screen resolution and then centers it.

**Form\_Open()**--Updates the menu bar and sets the MainMenu form to invisible so that the screen is easier to view.

**MoveToCenter()**--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by form basis.

## G. SUB SELECT MISHAP CLASS

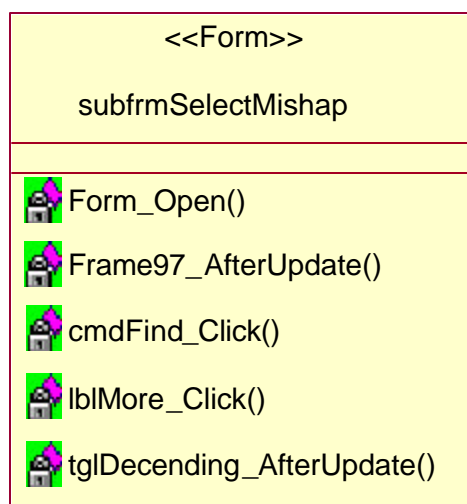


Figure E.7. Class Diagram for Sub Select Mishap Class.

### 1. Class Description

This class is used in a form/subform relationship with the SelectMishap form. It displays the mishaps in a sortable order.

### 2. Data Member Description

None.

### 3. Method Description

**Form\_Open()**--Sets color values for the columns in the form as well as initial sort order.

**Frame97\_AfterUpdate()**--Logic module that reacts to radio button clicks. Sorts the data on the form in the order specified.

**lblMore\_Click()**--Reacts to the click of the "More..." box in each row of the data in the form. Opens a form that displays a more detailed description of the mishap because these descriptions are too big to fit in the datagrid of the form.

**tglDecending\_AfterUpdate()**--Logic module that sorts the data on the form in ascending or descending order based on the state of the toggle button.

## H. EDIT MISHAP CLASS

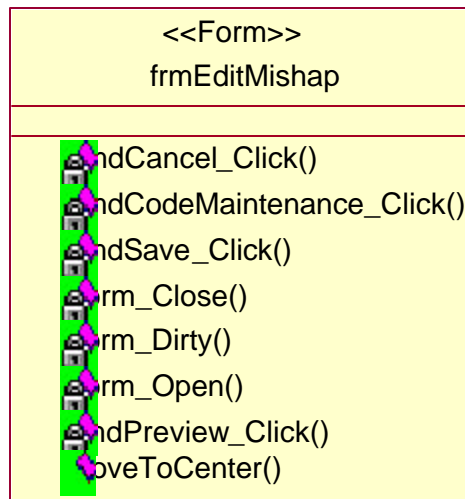


Figure E.8. Edit Mishap Class Diagram.

### **1. Class Description**

This class is used to edit mishaps and add factors. It is similar to the 2-0-1-2-subFrm-View mishaps class, but offers the additional capability to edit the data in the underlying tables.

### **2. Data Member Description**

None.

### **3. Method Description**

**cmdCancel\_Click()**--Closes the form undoing changes but only for events that have not already been refreshed. For example, if you add a factor, the entire form is refreshed . . . so clicking cancel cannot undo the addition of the factor - you have to use the delete button. This function is only capable of undoing actions made to controls in the top portion of the form, and then, only if a refresh has not yet been committed.

**cmdCodeMaintenance\_Click()**--Opens the code maintenance form.

**cmdSave\_Click()**--Saves the state of the data and closes the form.

**Form\_Close ()**--Closes the form.

**Form\_Dirty ()**--If changes are made to the mishap displayed in this form then the SelectMishap form will need to be updated when this form is closed. This function flags a global variable so that when the SelectMishap form is reactivated, it refreshes to display the changes.

**Form\_Load()**--Dynamically resizes the form to the users screen resolution and then centers it.

**Form\_Open()**--If this form is opened from the 1-0-0-5-frm-AddMishap then the record that was just added needs to be viewed in this form otherwise, it will display the record passed to it in the GlobalDeclarations.gLngMishapToGet global variable.

**cmdPreview\_Click()**--Opens the Mishap Snapshot report.

**MoveToCenter()**--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form

gets its own version of this function so that minor adjustments can be made on a form by form basis.

## I. MISHAP DESCRIPTION CLASS

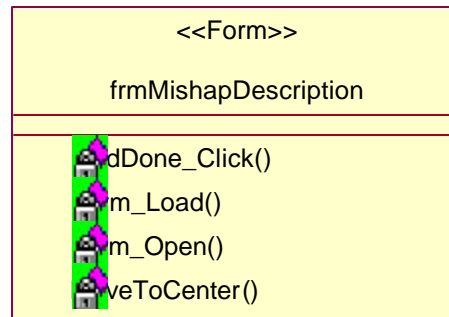


Figure E.9. Mishap Description Class Diagram.

### 1. Class Description

This class updates the menu bar and shows the value of the description for the mishap stored.

### 2. Data Member Description

None.

### 3. Method Description

**cmdDone\_Click ()**--Closes the form.

**Form\_Load ()**--Dynamically resizes the form to the users screen resolution and then centers it.

**Form\_Open ()**--Updates the menu bar and sets shows the value of the description for the mishap stored in the GlobalDeclarations.gStrDescription global variable.

**MoveToCenter()**--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by form basis.

## J. FACTORS CLASS

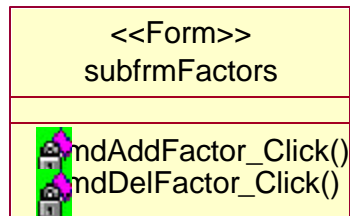


Figure E.10. Factors Class Diagram.

### 1. Class Description

This class is used in a form/subform relationship with the EditMishap form to display, add, and delete factors to a mishap.

### 2. Data Member Description

None.

### 3. Method Description

**cmdAddFactor\_Click ()**--Adds a blank factor to the mishap indicated by the `GlobalDeclarations.gLngMishapToGet` global variable.

**cmdDelFactor\_Click ()**--Deletes the factor with the current focus.

## K. ADD MISHAP CLASS

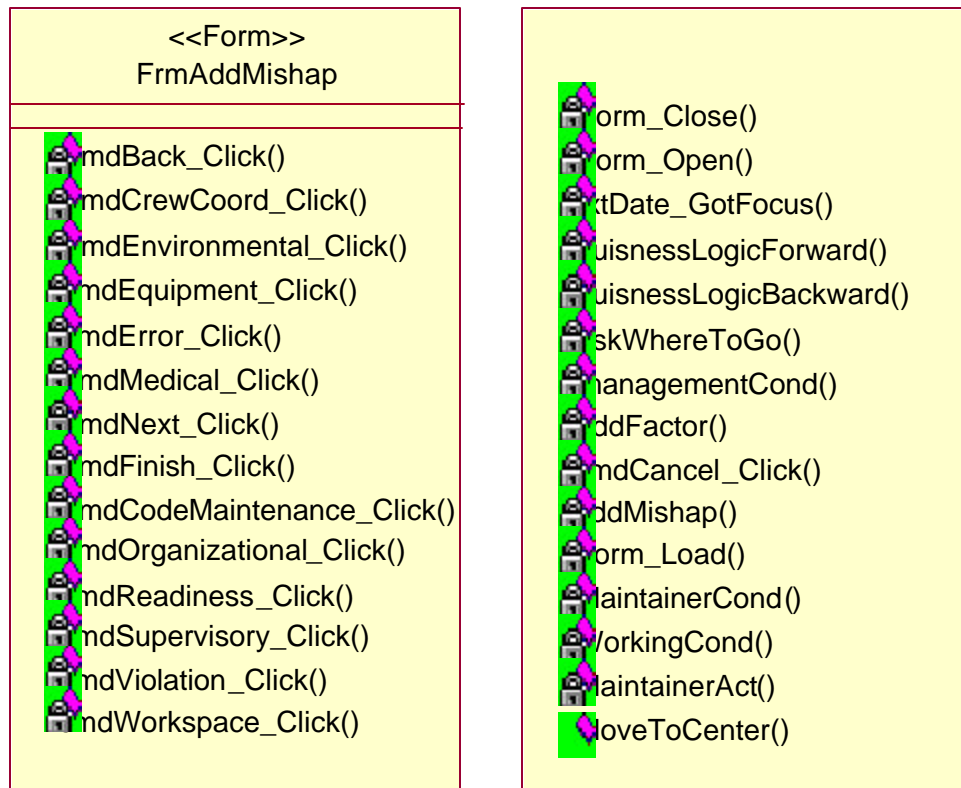


Figure E.11. Add Mishap Class Diagram.

### 1. Class Description

This class is a wizard used to add Mishaps to the database. The illusion of many forms is created using a TAB control on the form and setting the "tab style" property to "None". THIS IS IMPORTANT. The only way to edit the other pages of the tab control is to set the tab property to "Tabs" when the form is in design view and then change it back to "None" when finished. If you don't do this, you cannot edit any of the pages of the wizard except the first one. After a mishap is added, the EditMishap form is opened with the newly added Mishap selected for editing. This allows the user to immediately add Factors without having to go back to the main menu.

### 2. Data Member Description

None.

### 3. Method Description

**cmdBack\_Click ()**--Switches form focus back one tab in the tab view control.



**cmdNext\_Click ()**--Switches form focus forward one tab in the tab view control.

**cmdFinish\_Click ()**--Adds the mishap to the database and opens the edit form so that the user can add factors.

**cmdCodeMaintenance\_Click ()**--Opens the code maintenance form.

**cmdCrewCoord\_Click ()**--For controlling movement between pages not capable of movement using the "next" function.

**cmdEnvironmental\_Click ()**--For controlling movement between pages not capable of movement using the "next" function.

**cmdEquipment\_Click ()**--For controlling movement between pages not capable of movement using the "next" function.

**cmdError\_Click ()**--For controlling movement between pages not capable of movement using the "next" function.

**cmdMedical\_Click ()**--For controlling movement between pages not capable of movement using the "next" function.

**cmdOrganizational\_Click ()**--For controlling movement between pages not capable of movement using the "next" function.

**cmdReadiness\_Click ()**--For controlling movement between pages not capable of movement using the "next" function.

**cmdSupervisory\_Click ()**--For controlling movement between pages not capable of movement using the "next" function.

**cmdViolation\_Click ()**--For controlling movement between pages not capable of movement using the "next" function.

**cmdWorkspace\_Click ()**--For controlling movement between pages not capable of movement using the "next" function.

**Form\_Close ()**--Closes the form.

**Form\_Load ()**--Dynamically resizes the form to the users screen resolution and then centers it.

**Form\_Open ()**--Initializes all variables.

**txtDate\_GotFocus ()**--Ensures date fields are properly formatted to medium date.

**businessLogicForward ()**--Logic to determine what page to go in the forward direction.

**businessLogicBackward ()**--Logic to determine what page to go in the Reverse direction.

**askWhereToGo ()**--Logic to determine what page to go to based on user input.

**managementCond ()**--For prompting users for type of 1st level factor to input.

**maintainerCond ()**--For prompting users for type of 1st level factor to input.

**workingCond ()**--For prompting users for type of 1st level factor to input.

**maintainerAct ()**--For prompting users for type of 1st level factor to input.

**addFactor ()**--Creates a new default factor.

**cmdCancel\_Click ()**--Closes the form undoing changes.

**addMishap ()**--Creates a new default Mishap.

**MoveToCenter ()**--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by form basis.

## L. CODE MAINTENANCE CLASS

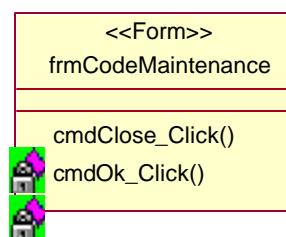


Figure E.12. Code Maintenance Class Diagram.

### 1. Class Description

Allows an Administrator to add codes directly to the database code lookup tables.

### 2. Data Member Description

None.

### 3. Method Description

**cmdClose\_Click()**--Closes the form.

**cmdOk\_Click()**--Opens the appropriate table for direct editing based on the radio button selection in the frame.

## M. CLOSE COMMAND CLASS

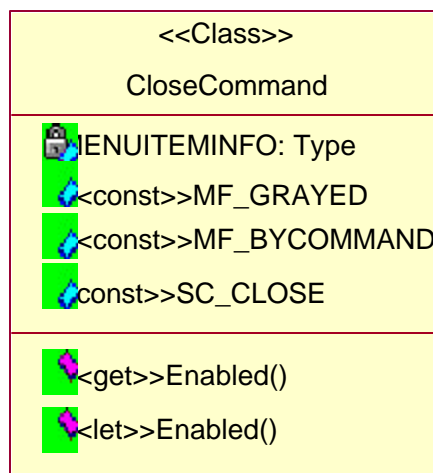


Figure E.13. Close Command Class Diagram.

### 1. Class Description

Disables the Access close button on the base Access application window.

### 2. Data Member Description

**MENUITEMINFO**--MENUITEMINFO structure used for API calls.

**MF\_GRAYED**--Constant value holding the value H1.

**MF\_BYCOMMAND**--Constant value holding the value H0.

**SC\_CLOSE**--Constant value holding the value HF060.

### 3. Method Description

**<<get>> Enabled ()**--Grays out the close button on the Access window.

**<<let>> Enabled ()**--Grays out the close button on the Access window.

## N. CONNECTION FUNCTIONS CLASS

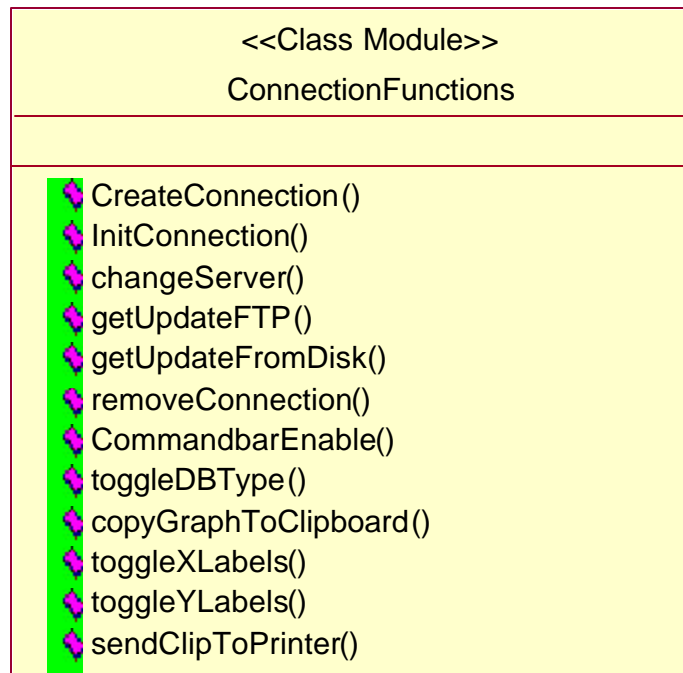


Figure E.14. Connection Functions Class Diagram.

### 1. Class Description

This module contains the vast majority of the "helper" functions used by the program. It contains functions for connecting and disconnecting the application to a SQL server, replacing the database via FTP and disk file, toggling database type, printing the MS Chart graphs from the windows clipboard, as well as, all command bar functions and command bar menu scripts.

### 2. Data Member Description

None.

### 3. Method Description

**CreateConnection ()**--Connects the application to a SQL server and provides the interface for the HFACS.dll. Read the initial values for most global program variables from the HFACS.ini file via the HFACS.dll and the SQL Server that becomes connected. Verifies the database type and ensure that the Server being connected to is of the proper type (military vice civilian).

**InitConnection ()**--Disables the Access "close" button on the main access window, preventing users from improperly shutting down the application. Launches the "Please Wait" form while the connection to the SQL server is initialized, giving the illusion of separate threads of execution and providing the user a screen to look at during this long process.

**ChangeServer()**--Provides the functionality to change server connections via the HFACS.dll.

**getUpdateFTP()**--Provides the functionality replace the database on the local SQL server via an FTP process. The user must be logged on with the sa account, being an administrator is not enough.

**GetUpdateFromDisk()**--Provides the functionality replace the database on the local SQL server via an file on a CD or network share process. The user must be logged on with the sa account, being an administrator is not enough.

**removeConnection()**--Properly disconnects the application from the SQL server and terminates the Access session.

**CommandbarEnable()**--Allows manipulation of command (menu bars). This function has four arguments: Cmdbar is a CommmandBar object that represents the command bar containing the menu item to be enabled or disabled. CmdBarEnabled is a Boolean value in which you pass "True" or "False" in order to enable or disable the menu item being manipulated. TopLevel is an integer representing the index of the Top-level menu item being manipulated. Sublevel is an optional integer representing the index of the menu item being manipulated under the Top-level menu item.

**toggleDBType()**--Properly disconnects the application from the SQL server and terminates the Access session.

**copyGraphToClipboard()**--Copies the MS Chart object on form TheActualGraph to the windows clipboard.

**toggleXLabels()**--Toggles the X axis values visible/hidden for the MS Chart object on form TheActualGraph.

**toggleYLabels()**--Toggles the Y axis values visible/hidden for the MS Chart object on form TheActualGraph.

**sendClipToPrinter()**--Prints the MS Chart object on form TheActualGraph.

## O. PLEASE WAIT CLASS

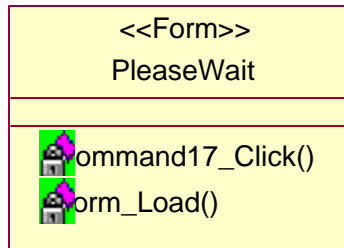


Figure E.15. Please Wait Class Diagram.

### 1. Class Description

This class is the splash screen that user sees at program initiation. It is responsible for setting global properties for the session at startup.

### 2. Data Member Description

None.

### 3. Method Description

**Command17\_Click ()**--Closes the form. This button is not visible during normal program operation and must be turned on in design view to use it. It is provided for troubleshooting connection problems which often result in a "hang" at this screen with now way to terminate program execution unless this button is enabled.

**Form\_Load ()**--Sets the global properties for the session. This includes application icon, margins, and other default behaviors.

## P. MAIN MENU CLASS

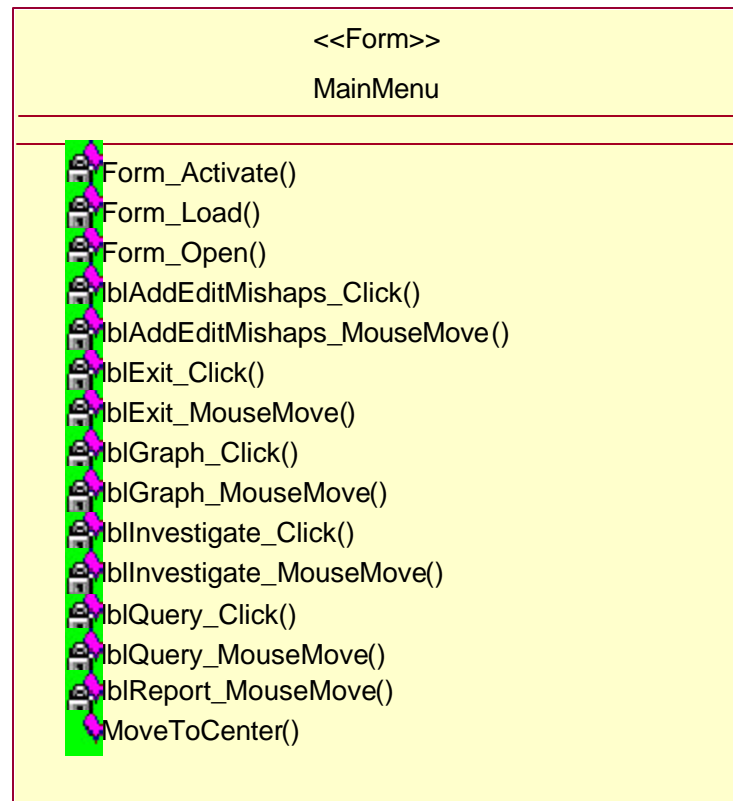


Figure E.16. Main Menu Class Diagram.

### 1. Class Description

This class is the main switchboard for the program. It is responsible for launching all other processes, connecting to the SQL server, validating Administrator settings, and determining O/S platform.

### 2. Data Member Description

None.

### 3. Method Description

**Form\_Activate ()**--Update the menu bar.

**Form\_Load ()**--Dynamically resizes the form to the users screen resolution and then centers it.

**Form\_Open()**--Set initial screen colors, determine OS type, and initiate connection to the SQL Server.

**lblAddEditMishaps\_Click()**--Only Administrators can access the administration functions and then, only for the local machine. This function ensures that the user is a Window O/S Administrator, a SQL Server Administrator, and an HFACS Administrator. If all these tests are passed, then the SelectMishap form is opened.

**lblAddEditMishaps\_MouseMove()**--Sets command button text colors.

**lblExit\_Click()**--Closes the program and properly disconnects from the SQL server.

**lblExit\_MouseMove()**--Sets command button text colors.

**lblGraph\_Click()**--Opens the Expert graph form (form-ExpertGraph).

**lblGraph\_MouseMove()**--Sets command button text colors.

**lblInvestigate\_Click()**--Launches the Invetigate.mdb Access database in a separate process.

**lblInvestigate\_MouseMove()**--Sets command button text colors.

**lblQuery\_Click()**--Opens the Expert graph form (form-QueryMenu).

**lblQuery\_MouseMove()**--Sets command button text colors.

**lblReport\_MouseMove()**--Sets command button text colors.

**MoveToCenter()**--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by form basis.



## Q. ACTUAL GRAPH CLASS

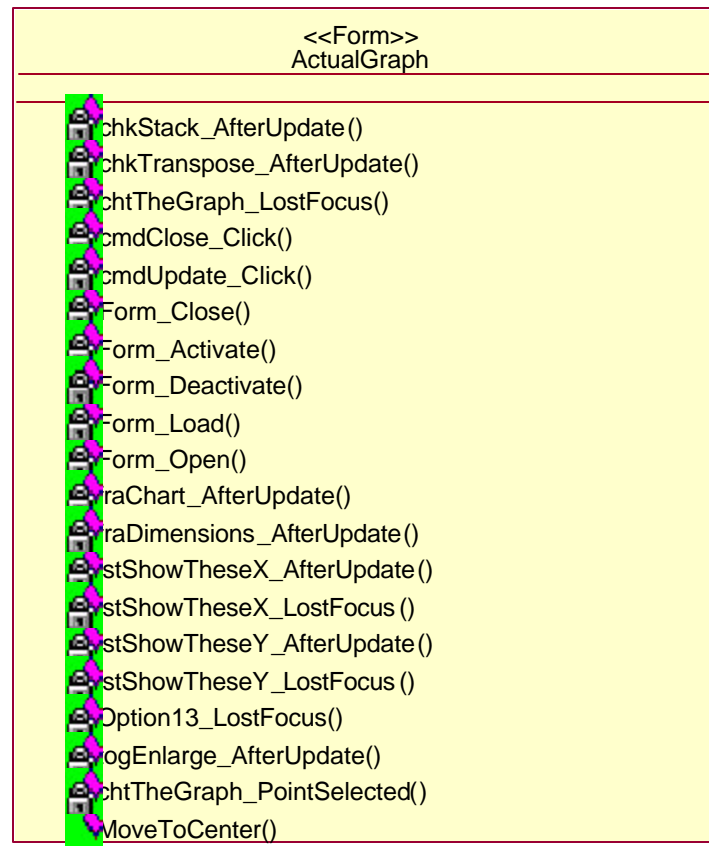


Figure E.17. Actual Graph Class Diagram.

### 1. Class Description

Uses the MSChart20 Active-X control to create a graph based upon global variables passed from the ExpertGraph form. The MSChart20 control creates a graph based upon values in its DataGrid. The datagrid is not visible and must be populated completely via code. Various methods in this class are used to populate the datagrid and then show portions of it based on input from the user. The datagrid data is obtained from the RAC (Replacement For Access Crosstab) stored procedures to create the crosstab results based on the values of GlobalDeclarations.gStrXFieldToGraph and GlobalDeclarations.gStrYFieldToGraph

### 2. Data Member Description

None.

### 3. Method Description

**chkStack\_AfterUpdate ()**--Sets the Stacking option of the MSChart control in response to a checkbox update.

**chkTranspose\_AfterUpdate ()**--Sets the DataSeriesInRow option of the MSChart control in response to a checkbox update.

**chtTheGraph\_LostFocus ()**--Updates the "Tips" label with information for the user.

**cmdClose\_Click ()**--Closes the form.

**cmdUpdate\_Click ()**--Rebuilds the MSChart20 control's Datagrid based upon `lstShowTheseX_AfterUpdate()` and `lstShowTheseY_AfterUpdate()` information (which corresponds to the users selections in the X and Y axis list box selection criteria).

**Form\_Close ()**--Closes the form.

**Form\_Activate ()**--Update the menu bar.

**Form\_Deactivate ()**--Updates the menu bar.

**Form\_Load ()**--Dynamically resizes the form to the users screen resolution and then centers it.

**Form\_Open ()**--Builds the MSChart20 control's Datagrid based upon the results of a RAC stored procedure (4-0-1-0-flanCrossTabForGraphing). Also, sets up visual aspects of the graph and populates the X and Y multi-select listboxes with values.

**fraChart\_AfterUpdate ()**--Sets the ChartType option of the MSChart control in response to a radio button selection. It has to check the value of `fraDimensions` to do this, so it knows if the chart should be 2d or 3d.

**fraDimensions\_AfterUpdate ()**--Sets the ChartType option with respect to number of dimensions (2d or 3d) of the MSChart control in response to a radio button selection. It has to check the value of `fraChartType` to do this, so it knows what style chart to create.

**lstShowTheseX\_AfterUpdate ()**--Builds the array used by cmdUpdate\_Click() to update the datagrid rows (X Axis) based on the users X-axis selections.

**lstShowTheseY\_AfterUpdate ()**--Builds the array used by cmdUpdate\_Click() to update the datagrid columns (Y Axis) based on the users Y-axis selections.

**lstShowTheseY\_LostFocus ()**--Updates the "Tips" label with information for the user.

**lstShowTheseX\_LostFocus ()**--Updates the "Tips" label with information for the user.

**Option13\_LostFocus()**--Updates the "Tips" label with information for the user.

**togEnlarge\_AfterUpdate()**--Enlarges or shrinks the form using the ezSizeForm class.

**chtTheGraph\_PointSelected()**--Updates the "Tips" label with information specified when the user clicks on a data point in the MSChart20 object.

**MoveToCenter()**--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by form basis.

## R. EXPERT GRAPH CLASS

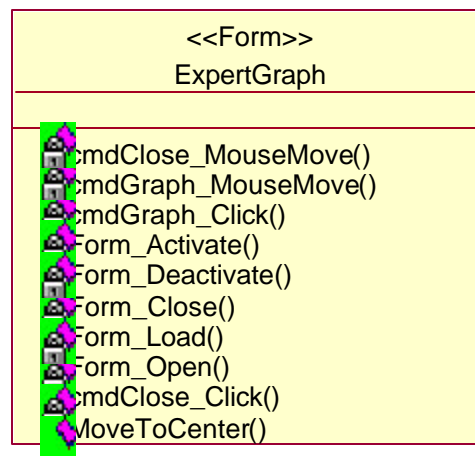


Figure E.18. Expert Graph Class Diagram.

### **1. Class Description**

This class is used to select the X and Y axis criteria and pass the users selections to global variables that the form TheActualGraph can use to display the graph.

### **2. Data Member Description**

None.

### **3. Method Description**

**cmdClose\_MouseMove ()**--Changes the color of the command button text in response to a mouse move event.

**cmdGraph\_MouseMove ()**--Changes the color of the command button text in response to a mouse move event.

**cmdGraph\_Click ()**--Passes the appropriate field names corresponding to user choices for X and Y axis graph criteria to global variables for the TheActualGraph form to actually create the graph.

**Form\_Activate ()**--Update the menu bar.

**Form\_Deactivate ()**--Updates the menu bar.

**Form\_Close ()**--Closes the form.

**Form\_Load ()**--Dynamically resizes the form to the users screen resolution and then centers it.

**Form\_Open ()**--Updates the menu bar and sets the focus to the close button.

**cmdClose\_Click ()**--Closes the form.

**MoveToCenter()**--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by form basis.

## S. SUMMARY CLASS

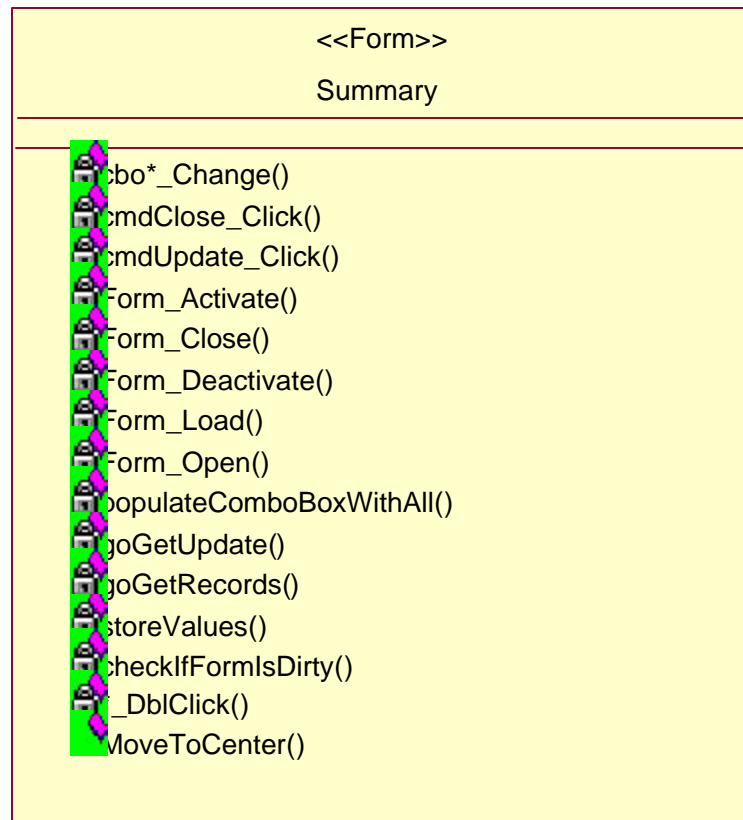


Figure E.19. Summary Class Diagram.

### 1. Class Description

This class is used to depict the table of factor vs. mishap counts and percentages. It allows the user to select criteria from combo boxes and fills then calculates the values for the table when the user clicks update.

When the form opens, it populates the combo boxes by running UNION queries to build the recordsets needed to serve as control sources. This is necessary to add the "<All>" choice. The only exception is the "Year" combo box. It uses a string manipulation function called populateComboBoxWithAll() to build a value list. This is necessary because the UNION method will only work with non-integer data types. The problem with the populateComboBoxWithAll() method is that it is limited in size to about 50 two dimensional entries. In addition, commas and semi-colons create problems and must be removed from the string during build.

Finally, when the user clicks double clicks a label in the table, code is executed that builds the input string for stored procedure `flanCountflanFilteredMishaps` which is the recordsource for the `ViewMishaps` form. this input string is then passed to the "view" form via a global variable and the `viewMishaps` form is opened.

## **2. Data Member Description**

None.

## **3. Method Description**

**cbo\*\_Change ()**--Used to mark the form as dirty (needing an update).Saves the state of the data (and size of the form). Applies to all methods that start with `cbo` and ends with `_Change`.

**cmdClose\_Click ()**--Closes the form.

**cmdUpdate\_Click()**--Updates all data on the form by calling `goGetUpdate()`.

**Form\_Activate()**--Update the menu bar.

**Form\_Close()**--Closes the form.

**Form\_Deactivate()**--Update the menu bar.

**Form\_Load()**--Dynamically resizes the form to the users screen resolution and then centers it.

**Form\_Open()**--Populates combo boxes. In order to allow the combo boxes to offer <All> as a choice, 2 methods are needed -- one for integers and another for strings. The `populateComboBoxWillAll()` subroutine is used for integers (like the Mishap Year), while stored procedures are used for strings.

Important to note that the `populateComboBoxWillAll()` will not work for creating strings of more than about 50 entries because the combo box rejects them as too, long. Stored procedures, however, do not suffer from this limitation.

**populateComboBoxWithAll()**--Makes a connection to the stored procedure passed-in and builds an string that can be used by a combo box to display and "iNumberColToGet" column drop down list. It has to check every record for commas and semi-colons in the data because the combo box interprets these two characters as

delimiters, so they must be replaced with some other character (a "-" is what we are using here).

**goGetUpdate()**--Builds the input string to pass based on the users combo box selection and uses this information to query again the underlying recordsource for this form. This updates the table to show the counts corresponding to the user's combo box criteria.

**goGetRecords()**--Builds the input string to pass to the stored procedure to get the correct records. Order of these if statements must match the SP. If <All> was selected, then pass " so that the SP knows the value is NULL. Once the input string is built, the 2-0-1-2-frm-ViewMishaps form is opened.

**storeValues()**--Store the values of the filter boxes on form open and after every update so that you have something to compare current values to. This way, you can trap when users make changes.

**checkIfFormIsDirty()**--If the user changed values in the combo boxes but has not updated the form, tell him about it and give the option to refresh before viewing records. If you don't do this, then the user might change the combo box criteria and then forget to hit the update button before double-clicking one of the boxes. This could create confusing results.

**\*\_DbfClick()**--Private subs-for detecting box double clicks follow. Three subroutines are needed for each box. One for the label and one for each text box (number and percentage). This applies to all functions that has \_DbfClick on its name.

**MoveToCenter()**--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by form basis.

## T. VIEW MISHAPS CLASS

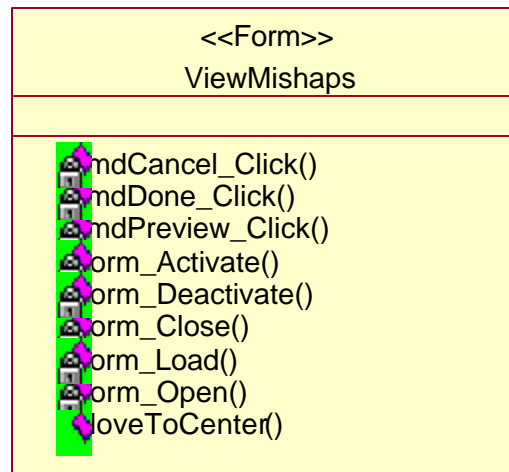


Figure E.20. View Mishaps Class Diagram.

### 1. Class Description

This class is used to view the mishaps with factors. It does not allow input, edit, or deletion of data. It is called by both the ExpertQueryForm and the Summary form. Because it is called by two different forms, it has the capability to determine which stored procedure to use as a record source based on the value of the GlobalDeclarations.bUseHFACSSummaryQuery global variable.

### 2. Data Member Description

None.

### 3. Method Description

**cmdCancel\_Click ()**--Saves the state of the data (and size of the form) and closes the form.

**cmdDone\_Click ()**--Closes the form.

**Form\_Activate ()**--Update the menu bar.

**Form\_Close ()**--Resets the flag used to tell the form which stored procedure to use for a record source.

**Form\_Deactivate ()**--Update the menu bar.



**Form\_Load ()**--Dynamically resizes the form to the users screen resolution and then centers it.

**Form\_Open()**--Determines which stored procedure to use as a record source based on the value of the GlobalDeclarations.bUseHFACSSummaryQuery global variable.

**cmdPreview\_Click ()**--If this program is being run with full-blown Access, this function opens the Mishap report. If it is being run with Runtime Access, then there is no support for reports and an error message is displayed.

**MoveToCenter()**--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by form basis.

## U. EXPERT QUERY CLASS

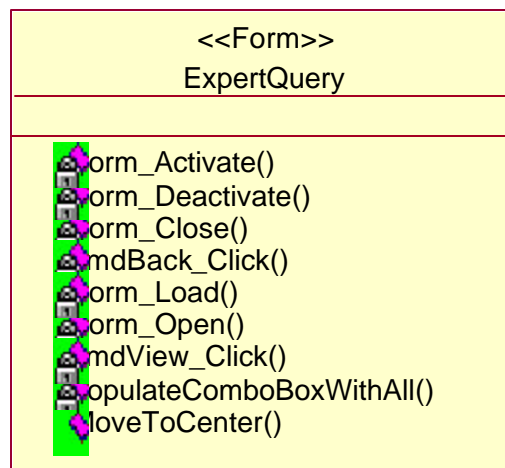


Figure E.21. Expert Query Class Diagram.

### 1. Class Description

This form allows the user to choose multiple criteria from a series of combo boxes and then query the database to open the ViewMishaps form and display the mishaps and factors.

When the form opens, it populates the combo boxes by running UNION queries to build the recordsets needed to serve as control sources. This is necessary to add the

"<All>" choice. The only exception is the "Year" combo box. It uses a string manipulation function called `populateComboBoxWithAll()` to build a value list. This is necessary because the UNION method will only work with non-integer data types. The problem with the `populateComboBoxWithAll()` method is that it is limited in size to about 50 two dimensional entries. In addition, commas and semi-colons create problems and must be removed from the string during build.

Finally, when the user clicks "View", code is executed that builds the input string for stored procedure `flanCountflanFilteredMishaps` which is the recordsource for the ViewMishaps form. This input string is then passed to the "view" form via a global variable and the viewMishaps form is opened.

## **2. Data Member Description**

None.

## **3. Method Description**

**Form\_Activate ()**--Update the menu bar.

**Form\_Deactivate ()**--Update the menu bar.

**Form\_Close ()**--Updates the menu bar.

**cmdBack\_Click ()**--Closes the form.

**Form\_Load ()**--Dynamically resizes the form to the users screen resolution and then centers it.

**Form\_Open()**--Populates combo boxes. In order to allow the combo boxes to offer <All> as a choice, 2 methods are needed -- one for integers and another for strings. The `populateComboBoxWillAll()` subroutine is used for integers (like the Mishap Year), while stored procedures are used for strings. Important to note that the `populateComboBoxWillAll()` will not work for creating strings of more than about 50 entries because the combo box rejects them as too, long. Stored procedures, however, do not suffer from this limitation.

**cmdView\_Click ()**--Builds the input string to pass to the stored procedure to get the correct records. Order of these if statements must match the SP. If <All> was selected, then pass " so that the SP knows the value is NULL. Once the input string is

built, a stored procedure is run from within this function to determine if there are actually any records in the database matching the users selections. If no records match, an error message is displayed. Otherwise the 2-0-1-2-frm-ViewMishaps form is opened.

**populateComboBoxWithAll ()**--Makes a connection to the stored procedure passed-in and builds a string that can be used by a combo box to display and "iNumberColToGet" column drop down list. It has to check every record for commas and semi-colons in the data because these two characters are interpreted by the combo box as delimiters, so they must be replaced with some other character (a "-" is what we are using here).

**MoveToCenter()**--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by form basis.

## V. QUERY MENU CLASS

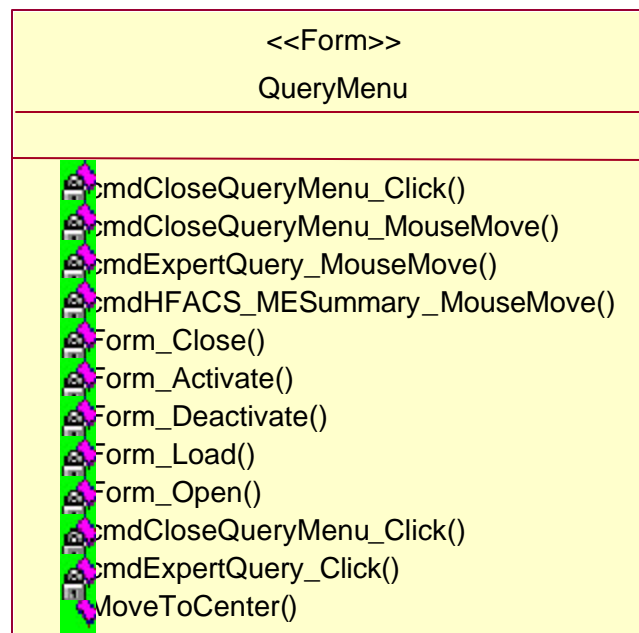


Figure E.22. Query Menu Class Diagram.

### **1. Class Description**

This class is the form for selecting the type of query to run. It has no special functionality or recordsource.

### **2. Data Member Description**

None.

### **3. Method Description**

**cmdCloseQueryMenu\_Click ()**--Closes the form.

**cmdCloseQueryMenu\_MouseMove ()**--Update text color on the command buttons in response to mouse over events.

**cmdExpertQuery\_MouseMove ()**--Update text color on the command buttons in response to mouse over events.

**cmdHFACS\_MESummary\_MouseMove ()**--Update text color on the command buttons in response to mouse over events.

**Form\_Close ()**--Closes the form.

**Form\_Activate ()**--Update the menu bar.

**Form\_Deactivate ()**--Update the menu bar.

**Form\_Load ()**--Dynamically resizes the form to the users screen resolution and then centers it.

**Form\_Open ()**--Updates the menu bar and sets the focus to the first command button, setting its color to blue.

**cmdCloseQueryMenu\_Click()**--Opens the Summary form.

**cmdCloseQueryMenu\_Click()**--Opens the ExpertQueryForm form.

**MoveToCenter()**--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by form basis.

## W. REPORT CLASS

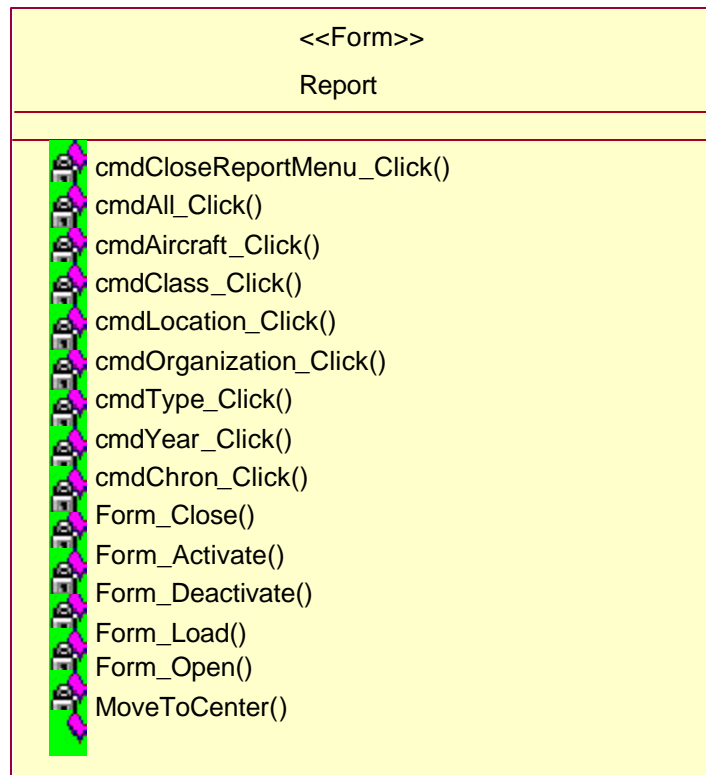


Figure E.23. Report Class Diagram.

### 1. Class Description

This class is the form for selecting the type of report to run.

### 2. Data Member Description

None.

### 3. Method Description

cmdCloseReportMenu\_Click ()--Closes the form.

cmdAll\_Click ()--Launch the report for all field values in response to command button click event.

cmdAircraft\_Click ()--Launch the report for sorting by aircraft reports in response to command button click event.

cmdClass\_Click ()--Launch the report for sorting by Class reports in response to command button click event.

cmdLocation\_Click ()--Launch the report for sorting by location reports in response to command button click event.

cmdOrganization\_Click ()--Launch the report for sorting by organization reports in response to command button click event.

cmdType\_Click ()--Launch the report for sorting by type reports in response to command button click event.

cmdYear\_Click ()--Launch the report for sorting by year reports in response to command button click event.

cmdChron\_Click ()--Launch the report for sorting by chronology reports in response to command button click event.

Form\_Close ()--Closes the form.

Form\_Activate ()--Update the menu bar.

Form\_Deactivate()--Update the menu bar.

Form\_Load()--Dynamically resizes the form to the users screen resolution and then centers it.

Form\_Open()--Updates the menu bar and sets the focus to the first command button, setting its color to blue.

MoveToCenter()--Centers the form on the screen. Using the ezSizeForm class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by form basis.

## APPENDIX F. BUSINESS LOGIC COMPONENT CODE

### Class-clForm Window

Option Compare Database  
Option Explicit

\*\*\*\*\*  
' Type declarations  
\*\*\*\*\*

Private Type RECT     'RECT structure used for API calls.  
    Left As Long  
    Top As Long  
    Right As Long  
    Bottom As Long  
End Type

Private Type POINTAPI 'POINTAPI structure used for API  
calls.  
    X As Long  
    Y As Long  
End Type

\*\*\*\*\*  
' Member variables  
\*\*\*\*\*

Private m\_hWnd As Long     'Handle of the window.  
Private m\_rectWindow As RECT 'Rectangle describing the  
sides of the last polled location of the window.

\*\*\*\*\*  
' Private error constants for use with RaiseError procedure  
\*\*\*\*\*

Private Const m\_ERR\_INVALIDHWND = 1  
Private Const m\_ERR\_NOPARENTWINDOW = 2

\*\*\*\*\*  
' API function declarations  
\*\*\*\*\*

Private Declare Function apiIsWindow Lib "user32" Alias  
"IsWindow" (ByVal hwnd As Long) As Long

Private Declare Function apiMoveWindow Lib "user32"  
Alias "MoveWindow" (ByVal hwnd As Long, ByVal X As  
Long, ByVal Y As Long, \_  
    ByVal nWidth As Long, ByVal nHeight As Long, ByVal  
bRepaint As Long) As Long  
    'Moves and resizes a window in the coordinate system of  
its parent window.

Private Declare Function apiGetWindowRect Lib "user32"  
Alias "GetWindowRect" (ByVal hwnd As Long, lpRect As  
RECT) As Long  
    'After calling, the lpRect parameter contains the RECT  
structure describing the sides of the window in screen  
coordinates.

Private Declare Function apiScreenToClient Lib "user32"  
Alias "ScreenToClient" (ByVal hwnd As Long, lpPoint As  
POINTAPI) As Long  
    'Converts lpPoint from screen coordinates to the  
coordinate system of the specified client window.

Private Declare Function apiGetParent Lib "user32" Alias  
"GetParent" (ByVal hwnd As Long) As Long  
    'Returns the handle of the parent window of the specified  
window.

#####  
'           CLASS DESCRIPTION  
#####  
'Class Name: clFormWindow.bas  
'  
'Author: Pat Flanders & Scott Tufts  
'

'Description: Moves and resizes a window in the coordinate  
system  
'of its parent window.

'References: None  
'

#####

\*\*\*\*\*  
'           FUNCTIONS  
\*\*\*\*\*

=====

'Function/Sub Name: RaiseError()  
'

'Description: Raises a user-defined error to the calling  
procedure.  
'

'Input: None  
'

'Output: None  
'

'References: None  
'

=====

Private Sub RaiseError(ByVal lngErrNumber As Long,  
ByVal strErrDesc As String)

    ERR.Raise vbObjectError + lngErrNumber,  
"clFormWindow", strErrDesc

End Sub

=====

'Function/Sub Name: UpdateWindowRect()  
'

'Description: Places the current window rectangle position (in  
'pixels, in coordinate system of parent window) in  
m\_rectWindow.  
'

'Input: None  
'

'Output: None  
'

References: None

'=====

Private Sub UpdateWindowRect()

Dim ptCorner As POINTAPI

If m\_hWnd = 0 Or apiIsWindow(m\_hWnd) Then  
apiGetWindowRect m\_hWnd, m\_rctWindow  
'm\_rctWindow now holds window coordinates in screen  
coordinates.

If Not Me.Parent Is Nothing Then  
'If there is a parent window, convert top, left of  
window from screen coordinates to parent window  
coordinates.

With ptCorner  
.X = m\_rctWindow.Left  
.Y = m\_rctWindow.Top  
End With

apiScreenToClient Me.Parent.hwnd, ptCorner

With m\_rctWindow  
.Left = ptCorner.X  
.Top = ptCorner.Y  
End With

'If there is a parent window, convert bottom, right of  
window from screen coordinates to parent window  
coordinates.

With ptCorner  
.X = m\_rctWindow.Right  
.Y = m\_rctWindow.Bottom  
End With

apiScreenToClient Me.Parent.hwnd, ptCorner

With m\_rctWindow  
.Right = ptCorner.X  
.Bottom = ptCorner.Y  
End With

End If  
Else  
RaiseError m\_ERR\_INVALIDHWND, "The window  
handle " & m\_hWnd & " is no longer valid."  
End If

End Sub

'=====

' Public read-write properties follow

'=====

Public Property Get hwnd() As Long  
'Returns the value the user has specified for the window's  
handle.

If m\_hWnd = 0 Or apiIsWindow(m\_hWnd) Then  
hwnd = m\_hWnd  
Else  
RaiseError m\_ERR\_INVALIDHWND, "The window  
handle " & m\_hWnd & " is no longer valid."  
End If

End Property

Public Property Let hwnd(ByVal lngNewValue As Long)

'Sets the window to use by specifying its handle.  
'Only accepts valid window handles.

If lngNewValue = 0 Or apiIsWindow(lngNewValue) Then  
m\_hWnd = lngNewValue  
Else  
RaiseError m\_ERR\_INVALIDHWND, "The value  
passed to the hWnd property is not a valid window handle."  
End If

End Property

Public Property Get Left() As Long  
'Returns the current position (in pixels) of the left edge of the  
window in the coordinate system of its parent window.

If m\_hWnd = 0 Or apiIsWindow(m\_hWnd) Then  
UpdateWindowRect  
Left = m\_rctWindow.Left  
Else  
RaiseError m\_ERR\_INVALIDHWND, "The window  
handle " & m\_hWnd & " is no longer valid."  
End If

End Property

Public Property Let Left(ByVal lngNewValue As Long)  
'Moves the window such that its left edge falls at the position  
indicated  
'(measured in pixels, in the coordinate system of its parent  
window).

If m\_hWnd = 0 Or apiIsWindow(m\_hWnd) Then  
UpdateWindowRect  
With m\_rctWindow  
apiMoveWindow m\_hWnd, lngNewValue, .Top,  
.Right - .Left, .Bottom - .Top, True  
End With  
Else  
RaiseError m\_ERR\_INVALIDHWND, "The window  
handle " & m\_hWnd & " is no longer valid."  
End If

End Property

'=====

Public Property Get Top() As Long  
'Returns the current position (in pixels) of the top edge of the  
window in the coordinate system of its parent window.

If m\_hWnd = 0 Or apiIsWindow(m\_hWnd) Then  
UpdateWindowRect  
Top = m\_rctWindow.Top  
Else  
RaiseError m\_ERR\_INVALIDHWND, "The window  
handle " & m\_hWnd & " is no longer valid."  
End If

End Property

Public Property Let Top(ByVal lngNewValue As Long)  
'Moves the window such that its top edge falls at the position  
indicated



'(measured in pixels, in the coordinate system of its parent window).

```

    If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
        UpdateWindowRect
        With m_rctWindow
            apiMoveWindow m_hWnd, .Left, lngNewValue,
            .Right - .Left, .Bottom - .Top, True
        End With
    Else
        RaiseError m_ERR_INVALIDHWND, "The window
        handle " & m_hWnd & " is no longer valid."
    End If

```

End Property

'-----

Public Property Get Width() As Long  
'Returns the current width (in pixels) of the window.

```

    If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
        UpdateWindowRect
        With m_rctWindow
            Width = .Right - .Left
        End With
    Else
        RaiseError m_ERR_INVALIDHWND, "The window
        handle " & m_hWnd & " is no longer valid."
    End If

```

End Property

Public Property Let Width(ByVal lngNewValue As Long)  
'Changes the width of the window to the value provided (in pixels).

```

    If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
        UpdateWindowRect
        With m_rctWindow
            apiMoveWindow m_hWnd, .Left, .Top,
            lngNewValue, .Bottom - .Top, True
        End With
    Else
        RaiseError m_ERR_INVALIDHWND, "The window
        handle " & m_hWnd & " is no longer valid."
    End If

```

End Property

'-----

Public Property Get Height() As Long  
'Returns the current height (in pixels) of the window.

```

    If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
        UpdateWindowRect
        With m_rctWindow
            Height = .Bottom - .Top
        End With
    Else
        RaiseError m_ERR_INVALIDHWND, "The window
        handle " & m_hWnd & " is no longer valid."
    End If

```

End Property

Public Property Let Height(ByVal lngNewValue As Long)  
'Changes the height of the window to the value provided (in pixels).

```

    If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
        UpdateWindowRect
        With m_rctWindow
            apiMoveWindow m_hWnd, .Left, .Top, .Right - .Left,
            lngNewValue, True
        End With
    Else
        RaiseError m_ERR_INVALIDHWND, "The window
        handle " & m_hWnd & " is no longer valid."
    End If

```

End Property

'=====

' Public read-only properties follow

Public Property Get Parent() As clFormWindow  
'Returns the parent window as a clFormWindow object.  
'For forms, this should be the Access MDI window.

```

    Dim fwParent As New clFormWindow
    Dim lngHWnd As Long

```

```

    If m_hWnd = 0 Then
        Set Parent = Nothing
    ElseIf apiIsWindow(m_hWnd) Then
        lngHWnd = apiGetParent(m_hWnd)
        fwParent.hwnd = lngHWnd
        Set Parent = fwParent
    Else
        RaiseError m_ERR_INVALIDHWND, "The window
        handle " & m_hWnd & " is no longer valid."
    End If

```

Set fwParent = Nothing

End Property

## **CLASS-CloseCommand**

```
Option Compare Database
Option Explicit
#####
'          CLASS DESCRIPTION
#####
'Class Name: CloseCommand.bas
'
'Author: Pat Flanders & Scott Tufts. Adapted from the
Microsoft
knowledgebase.
'
'Description: Disables the Access close button on the base
Access
'application window.
'
'References: None
'
#####

*****
'          DECLARES
*****
Private Declare Function GetSystemMenu Lib "user32"
(ByVal hwnd As Long, _
ByVal bRevert As Long) As Long

Private Declare Function EnableMenuItem Lib "user32"
(ByVal hMenu As _
Long, ByVal wIDEnableItem As Long, ByVal wEnable As
Long) As Long

Private Declare Function GetMenuItemInfo Lib "user32"
Alias _
"GetMenuItemInfoA" (ByVal hMenu As Long, ByVal un
As Long, ByVal b As _
Long, lpMenuItemInfo As MENUITEMINFO) As Long

Private Type MENUITEMINFO
cbSize As Long
fMask As Long
fType As Long
fState As Long
wID As Long
hSubMenu As Long
hbmChecked As Long

hbmUnchecked As Long
dwItemData As Long
dwTypeData As String
cch As Long
End Type

Const MF_GRAYED = &H1&
Const MF_BYCOMMAND = &H0&
Const SC_CLOSE = &HF060&

*****
'          PUBLIC PROPERTIES
*****
Public Property Get Enabled() As Boolean
Dim hwnd As Long
Dim hMenu As Long
Dim result As Long
Dim MI As MENUITEMINFO

MI.cbSize = Len(MI)
MI.dwTypeData = String(80, 0)
MI.cch = Len(MI.dwTypeData)
MI.fMask = MF_GRAYED
MI.wID = SC_CLOSE
hwnd = Application.hWndAccessApp
hMenu = GetSystemMenu(hwnd, 0)
result = GetMenuItemInfo(hMenu, MI.wID, 0, MI)
Enabled = (MI.fState And MF_GRAYED) = 0
End Property

Public Property Let Enabled(boolClose As Boolean)
Dim hwnd As Long
Dim wFlags As Long
Dim hMenu As Long
Dim result As Long

hwnd = Application.hWndAccessApp
hMenu = GetSystemMenu(hwnd, 0)
If Not boolClose Then
wFlags = MF_BYCOMMAND Or MF_GRAYED
Else
wFlags = MF_BYCOMMAND And Not MF_GRAYED
End If
result = EnableMenuItem(hMenu, SC_CLOSE, wFlags)
End Property
```

## **FORMCLASS-1-0-0-1-frm-SelectMishap**

```
Option Compare Database
Option Explicit
#####
'          FORM DESCRIPTION
#####
'Class Name: 1-0-0-0-frm-SelectMishap
'
'Author: Pat Flanders & Scott Tufts
'
'This class displays all the Mishaps in the database and
allows the
'user to sort them by various fields in order to select a mishap
to view or edit. It has buttons that allow initiation of a new
'Mishap or deletion of an existing mishap.
'
'References:
'   - 1-0-0-1-subFrm-SelectMishap
'   - clFormWindow
'   - ez_SizingFunctions
'   - GlobalDeclarations
'
#####

*****
'          FUNCTIONS
*****

=====
'Function/Sub Name: cmdDone_Click()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub cmdDone_Click()

    DoCmd.Close acForm, "1-0-0-0-frm-SelectMishap"

End Sub

=====
'Function/Sub Name: cmdViewMishap_Click()
'
'Description: Opens the mishap selected in the subform.
'
'Input: None
'
'Output: None
'
'References: GlobalDeclarations.gLngMishapToGet is a
global variable
holding the value of the mishap ID
'
=====
Private Sub cmdViewMishap_Click()

    On Error GoTo errorHandler
    GlobalDeclarations.gLngMishapToGet =
Me.Manage_Mishaps.Form![MishapID]
    Me.TxtGlobalFocus.Value =
GlobalDeclarations.gLngMishapToGet
    Me.Visible = False

    DoCmd.OpenForm "1-0-0-2-frm-EditMishap"
    Exit Sub

errorHandler:
    DoCmd.Beep
    MsgBox "There are no Mishaps to select!", vbOKOnly +
vbExclamation, "Error"

End Sub

=====
'Function/Sub Name: cmdAdd_Click()
'
'Description: Opens the add mishap wizard.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub cmdAdd_Click()

    Me.Visible = False
    DoCmd.OpenForm "1-0-0-5-frm-AddMishap"

End Sub

=====
'Function/Sub Name: cmdKill_Click()
'
'Description: Deletes the mishap selected in the subform.
'
'Input: None
'
'Output: None
'
'References: GlobalDeclarations.gLngMishapToGet is a
global variable
holding the value of the mishap ID
'
=====
Private Sub cmdKill_Click()

    On Error GoTo errorHandler

    'Store the value of the mishap selected in the subform in a
'global variable.
    GlobalDeclarations.gLngMishapToGet =
Me.Manage_Mishaps.Form![MishapID]

    'Also, store it in a text box.
    Me.TxtGlobalFocus.Value =
GlobalDeclarations.gLngMishapToGet

errorHandler:
    DoCmd.Beep
    MsgBox "There are no Mishaps to select!", vbOKOnly +
vbExclamation, "Error"

End Sub
```

```

Dim response As Variant

DoCmd.Beep
response = MsgBox("You are about to permanently delete
the record for MISHAP #" & Me.TxtGlobalFocus.Value & "
and all its related Factors." & Chr(13) & Chr(13) & "It is
STRONGLY recommended that you do not delete mishaps
from the database because this removes all references of
them." & Chr(13) & Chr(13) & "Do you want to delete this
Mishap record despite this warning?", vbYesNo +
vbQuestion + vbDefaultButton2, "Permanently Delete
Mishap?")

If response = vbYes Then

'Declare objects for querying a stored procedure to get
the new record
Dim rsTheNewMishap As New Recordset
Dim commandADO As New ADODB.Command
Dim conADO As New ADODB.Connection

' This is where we create the Connection object.
Set conADO = CurrentProject.Connection

rsTheNewMishap.Open "DELETE tblMishaps WHERE
tblMishaps.MishapID=" & Me.TxtGlobalFocus.Value,
conADO, , , adCmdText

'Destroy objects used for the query
Set commandADO = Nothing
Set conADO = Nothing
Set rsTheNewMishap = Nothing

Me.Manage_Mishaps.Requery

End If

Exit Sub

ErrorHandler:

DoCmd.Beep
MsgBox "There are no MishapDates to delete!", vbOKOnly
+ vbExclamation, "Error"

End Sub

'=====
'Function/Sub Name: Form_Activate()
'
'Description: Update the menu bar and see if the subform
needs to
'be refreshed.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Activate()

'Disable database replacement if not logged-in as local.

```

```

Dim bTemp As Boolean
If GlobalDeclarations.gStrServerName = "(local)" Then
bTemp =
CommandBarEnable(CommandBars("mnuAdmin"), True, 2)
Else
bTemp =
CommandBarEnable(CommandBars("mnuAdmin"), False, 2)
End If
Application.CommandBars("mnuAdmin").Visible = True

'Refresh the form if returning from a process that made it
dirty.
If GlobalDeclarations.gFormNeedsRefresh = True Then
Me.Manage_Mishaps.Requery
GlobalDeclarations.gFormNeedsRefresh = False
End If

End Sub

'=====
'Function/Sub Name: Form_Close()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Close()

Application.CommandBars("mnuProgramMain").Visible =
True
Forms![MainMenu].Visible = True

End Sub

'=====
'Function/Sub Name: Form_Deactivate()
'
'Description: Updates the menu bar.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Deactivate()
Application.CommandBars("mnuAdmin").Visible = False
End Sub

'=====
'Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None

```

```

'
'References:
'   - ezSizeForm
'
'=====
Private Sub Form_Load()
    'Dynamically resize the form based on screen resolution.
    ezSizeForm Me, -1
    MoveToCenter "1-0-0-0-frm-SelectMishap"
End Sub

'=====
Function/Sub Name: Form_Open()
'
'Description: Updates the menu bar and sets the MainMenu
form to
'invisible so that the screen is easier to view.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Open(Cancel As Integer)

    'Disable database replacement if not logged-in as local.
    Dim bTemp As Boolean
    If GlobalDeclarations.gStrServerName = "(local)" Then
        bTemp =
CommandbarEnable(CommandBars("mnuAdmin"), True, 2)
    Else
        bTemp =
CommandbarEnable(CommandBars("mnuAdmin"), False, 2)
    End If
    Application.CommandBars("mnuAdmin").Visible = True

    Forms![MainMenu].Visible = False

    On Error Resume Next

    Me.TxtGlobalFocus.Value =
GlobalDeclarations.gLngMishapToGet

    DoCmd.GoToControl "Manage_Mishaps"
End Sub

' =====

```

```

Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'   - clFormWindow
'
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        .Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing
End Sub

'=====
Function/Sub Name: Label127_DblClick()
'
'Description: Easter Egg Code. No further explanation
provided.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Label127_DblClick(Cancel As Integer)
    DoCmd.OpenForm "EasterEgg"
End Sub

```

## **FORMCLASS-1-0-0-1-subfrm-SelectMishap**

```
Option Compare Database
Option Explicit
'#####
'          FORM DESCRIPTION
'#####
'Class Name: 1-0-0-1-subfrm-SelectMishap
'
'Author: Pat Flanders & Scott Tufts
'
'This class is used in a form/subform relationship with the
'1-0-0-0-firm-SelectMishap form. It displays the mishaps in a
'sortable order.
'
'References:
'   - clFormWindow
'   - ez_SizingFunctions
'   - GlobalDeclarations
'#####

*****
'          FUNCTIONS
*****

'=====
'Function/Sub Name: Form_Open()
'
'Description: Sets color values for the columns in the form as
'as initial sort order.
'
'Input: None
'
'Output: None
'
'References: None
'=====
Private Sub Form_Open(Cancel As Integer)

    Me.tglDecending.Value = 0
    Me.OrderBy = "[MishapDate] ASC"
    Me.MishapDate.ForeColor = RGB(10, 140, 50)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
    Me.Type_FK.ForeColor = RGB(0, 0, 0)
    Me.MishapID.ForeColor = RGB(0, 0, 0)

End Sub

'=====
'Function/Sub Name: Frame97_AfterUpdate()
'
'Description: Logic module that reacts to radio button clicks.
Sorts
'the data on the form in the order specified.
'
'Input: None
'
'Output: None
'
'References: None
'=====

Private Sub Frame97_AfterUpdate()

    If Me.Frame97 = 1 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[MishapDate] DESC"
        Else
            Me.OrderBy = "[MishapDate] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(10, 140, 50)
        Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
        Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If
    If Me.Frame97 = 2 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[OrgID_FK] DESC"
        Else
            Me.OrderBy = "[OrgID_FK] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(0, 0, 0)
        Me.OrgID_FK.ForeColor = RGB(10, 140, 50)
        Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If
    If Me.Frame97 = 3 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[Aircraft_FK] DESC"
        Else
            Me.OrderBy = "[Aircraft_FK] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(0, 0, 0)
        Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
        Me.Aircraft_FK.ForeColor = RGB(10, 140, 50)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If
    If Me.Frame97 = 4 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[Class_FK] DESC"
        Else
            Me.OrderBy = "[Class_FK] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(0, 0, 0)
        Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
        Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
        Me.Class_FK.ForeColor = RGB(10, 140, 50)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If
    If Me.Frame97 = 5 Then
        If Me.tglDecending.Value = -1 Then
```

```

        Me.OrderBy = "[MishapLocation] DESC"
    Else
        Me.OrderBy = "[MishapLocation] ASC"
    End If
    Me.MishapDate.ForeColor = RGB(0, 0, 0)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(10, 140, 50)
    Me.Type_FK.ForeColor = RGB(0, 0, 0)
    Me.MishapID.ForeColor = RGB(0, 0, 0)
End If
If Me.Frame97 = 6 Then
    If Me.tglDecending.Value = -1 Then
        Me.OrderBy = "[Type_FK] DESC"
    Else
        Me.OrderBy = "[Type_FK] ASC"
    End If
    Me.MishapDate.ForeColor = RGB(0, 0, 0)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
    Me.Type_FK.ForeColor = RGB(10, 140, 50)
    Me.MishapID.ForeColor = RGB(0, 0, 0)
End If
If Me.Frame97 = 7 Then
    If Me.tglDecending.Value = -1 Then
        Me.OrderBy = "[MishapID] DESC"
    Else
        Me.OrderBy = "[MishapID] ASC"
    End If
    Me.MishapDate.ForeColor = RGB(0, 0, 0)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
    Me.Type_FK.ForeColor = RGB(0, 0, 0)
    Me.MishapID.ForeColor = RGB(10, 140, 50)
End If

End Sub

'=====
'Function/Sub Name: lblMore_Click()
'
'Description: Reacts to the click of the "More..." box in each
'row
'of the data in the form. Opens a form that displays a more
'detailed
'description of the mishap because these descriptions are too
'big
'to fit in the datagrid of the form.
'
'Input: None
'
'Output: None
'
'References:
'    - 1-0-0-3-PopUpFrm-MishapDescription
'
'=====
Private Sub lblMore_Click()
    gStrDescription = Me.lblDescription.Value
    DoCmd.OpenForm "1-0-0-3-PopUpFrm-MishapDescription"

```

```

End Sub

'=====
'Function/Sub Name: tglDecending_AfterUpdate()
'
'Description: Logic module that sorts the data on the form in
'ascending or descending order based on the state of the toggle
button.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub tglDecending_AfterUpdate()

    If Me.Frame97 = 1 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[MishapDate] DESC"
        Else
            Me.OrderBy = "[MishapDate] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(10, 140, 50)
        Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
        Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If
    If Me.Frame97 = 2 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[OrgID_FK] DESC"
        Else
            Me.OrderBy = "[OrgID_FK] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(0, 0, 0)
        Me.OrgID_FK.ForeColor = RGB(10, 140, 50)
        Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If
    If Me.Frame97 = 3 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[Aircraft_FK] DESC"
        Else
            Me.OrderBy = "[Aircraft_FK] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(0, 0, 0)
        Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
        Me.Aircraft_FK.ForeColor = RGB(10, 140, 50)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If
    If Me.Frame97 = 4 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[Class_FK] DESC"
        Else
            Me.OrderBy = "[Class_FK] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(0, 0, 0)

```

```

Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
Me.Class_FK.ForeColor = RGB(10, 140, 50)
Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
Me.Type_FK.ForeColor = RGB(0, 0, 0)
Me.MishapID.ForeColor = RGB(0, 0, 0)
End If
If Me.Frame97 = 5 Then
    If Me.tglDecending.Value = -1 Then
        Me.OrderBy = "[MishapLocation] DESC"
    Else
        Me.OrderBy = "[MishapLocation] ASC"
    End If
    Me.MishapDate.ForeColor = RGB(0, 0, 0)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(10, 140, 50)
    Me.Type_FK.ForeColor = RGB(0, 0, 0)
    Me.MishapID.ForeColor = RGB(0, 0, 0)
End If
If Me.Frame97 = 6 Then
    If Me.tglDecending.Value = -1 Then
        Me.OrderBy = "[Type_FK] DESC"
    Else
        Me.OrderBy = "[Type_FK] ASC"
    End If

```

```

End If
Me.MishapDate.ForeColor = RGB(0, 0, 0)
Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
Me.Class_FK.ForeColor = RGB(0, 0, 0)
Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
Me.Type_FK.ForeColor = RGB(10, 140, 50)
Me.MishapID.ForeColor = RGB(0, 0, 0)
End If
If Me.Frame97 = 7 Then
    If Me.tglDecending.Value = -1 Then
        Me.OrderBy = "[MishapID] DESC"
    Else
        Me.OrderBy = "[MishapID] ASC"
    End If
    Me.MishapDate.ForeColor = RGB(0, 0, 0)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
    Me.Type_FK.ForeColor = RGB(0, 0, 0)
    Me.MishapID.ForeColor = RGB(10, 140, 50)
End If
End Sub

```



## **FORMCLASS-1-0-0-2-frm-EditMishap**

```
Option Compare Database
Option Explicit
#####
'          FORM DESCRIPTION
#####
'Class Name: 1-0-0-2-frm-EditMishap
'
'Author: Pat Flanders & Scott Tufts
'
'This class is used to edit mishaps and add factors. It is
similar
'to the 2-0-1-2-subFrm-View mishaps class, but offers the
additional
'capability to edit the data in the underlying tables.
'
'References:
'   - 1-0-0-7-PopUpFrm-CodeMaintenance
'   - 1-0-0-4-subFrm-Factors
'   - clFormWindow
'   - ez_SizingFunctions
'   - GlobalDeclarations
'
#####

*****
'          FUNCTIONS
*****

=====
'Function/Sub Name: cmdCancel_Click()
'
'Description: Closes the form undoing changes BUT ONLY
for events
'that have not already been refreshed. For example, if you
add
'a factor, the entire form is refreshed . . . so clicking cancel
'cannot undo the addition of the factor - you have to use the
'delete button. This function is only capable of undoing
actions
'made to controls in the top portion of the form, and then,
only
'if a refresh has not yet been committed.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub cmdCancel_Click()

On Error GoTo Err_CmdCancel_Click

    DoCmd.DoMenuItem acFormBar, acEditMenu, acUndo, ,
acMenuVer70
    DoCmd.Close

Exit_CmdCancel_Click:
Exit Sub

Err_CmdCancel_Click:
DoCmd.Close

End Sub

=====
'Function/Sub Name: cmdCodeMaintenance_Click()
'
'Description: Opens the code maintenance form.
'
'Input: None
'
'Output: None
'
'References:
'   - 1-0-0-7-PopUpFrm-CodeMaintenance
'
=====
Private Sub cmdCodeMaintenance_Click()
    DoCmd.OpenForm "1-0-0-7-PopUpFrm-
CodeMaintenance"
End Sub

=====
'Function/Sub Name: cmdSave_Click()
'
'Description: Saves the state of the data and closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub cmdSave_Click()

    On Error GoTo Err_Blanks:

    DoCmd.Requery

Exit_cmdSave:
DoCmd.Close
Exit Sub

Err_Blanks:
GoTo Exit_cmdSave

End Sub

=====
'Function/Sub Name: Form_Activate()
'
'Description: Update the menu bar.
'
'Input: None
'
'Output: None
'
'References: None
'
```

```

'=====
Private Sub Form_Activate()
    Application.CommandBars("mnuOther").Visible = True
End Sub

'=====
Function/Sub Name: Form_Close()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Close()
    Forms![1-0-0-0-fm-SelectMishap].Visible = True
End Sub

'=====
Function/Sub Name: Form_Deactivate()
'
'Description: Updates the menu bar.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Deactivate()
    Application.CommandBars("mnuOther").Visible = False
End Sub

'=====
Function/Sub Name: Form_Dirty()
'
'Description: If changes are made to the mishap displayed in
this form
'then the 1-0-0-0-fm-SelectMishap form will need to be
updated when
'this form is closed. This function flags a global variable so
that
'when the 1-0-0-0-fm-SelectMishap form is reactivated, it
refreshes
'to display the changes.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Dirty(Cancel As Integer)
    MsgBox "The form is now dirty"
    GlobalDeclarations.gFormNeedsRefresh = True
End Sub

'=====

```

```

Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'
'References:
'    - ezSizeForm
'
'=====
Private Sub Form_Load()
    ezSizeForm Me, -1
    MoveToCenter "1-0-0-2-fm-EditMishap"
End Sub

'=====
Function/Sub Name: Form_Open()
'
'Description: If this form is opened from the 1-0-0-5-fm-
AddMishap
'then the record that was just added needs to be viewed in this
form
'otherwise, it will display the record passed to it in the
'GlobalDeclarations.gLngMishapToGet global variable.
'Input: None
'
'Output: None
'
'References:
'    - GlobalDeclarations
'
'=====
Private Sub Form_Open(Cancel As Integer)

    Application.CommandBars("mnuOther").Visible = True

    'Set the unique table for the underlying stored procedure
with code
    'because it sometimes dissapears when using the visual
property sheet.
    Me.UniqueTable = "tblMishaps"

    'Check to see if you are coming here from the Add Mishap
Wizard or just
    'from the select mishap form.
    If GlobalDeclarations.gBlnAddAMishap = True Then
        DoCmd.Close acForm, "1-0-0-5-fm-AddMishap"
        GlobalDeclarations.gBlnAddAMishap = False

        'Declare objects for querying a stored procedure to get
the new record
        Dim rsTheNewMishap As New Recordset
        Dim commandADO As New ADODB.Command
        Dim conADO As New ADODB.Connection

        ' This is where we create the Connection object.
        Set conADO = CurrentProject.Connection

        'Figure out what record was just added
        rsTheNewMishap.Open "SELECT max(MishapID)
FROM tblMishaps", conADO, , , adCmdText
        rsTheNewMishap.MoveFirst

```

```

GlobalDeclarations.gLngMishapToGet =
rsTheNewMishap.Fields(0)

'Destroy objects used for the query
Set commandADO = Nothing
Set conADO = Nothing
Set rsTheNewMishap = Nothing

'Set the inputparameters for opening the form
Me.InputParameters = "@MishapID int= " &
GlobalDeclarations.gLngMishapToGet

'Set the Title in the form header
Me.txtTitle.Value = [MishapID] & " - " & [OrgName] &
" - " & [Aircraft_FK]
Else
'This is a normal edit (not an add)
'Set the inputparameters for opening the form
Me.InputParameters = "@MishapID int= " &
GlobalDeclarations.gLngMishapToGet

'Set the Title in the form header
Me.txtTitle.Value = [MishapID] & " - " & [OrgName] &
" - " & [Aircraft_FK]
End If

End Sub

'=====
'Function/Sub Name: cmdPreview_Click()
'Description: Opens the Mishap Snapshot report.
'Input: None
'Output: None
'References:
'      - 1-0-MishapSnapshot-OpenMishaps
'=====
Private Sub cmdPreview_Click()

    On Error GoTo startError

    Me.Refresh

    GlobalDeclarations.gLngMishapToGet = Me.txtMishapID

```

```

DoCmd.OpenReport "1-0-MishapSnapshot-
OpenMishaps", acViewPreview

exitSub:

Exit Sub

startError:

    MsgBox "You must have a default printer installed in
order to preview reports.", vbCritical + vbOKOnly, "Can't
Find A Printer"

End Sub

'=====
'Function/Sub Name: MoveToCenter()
'Description: Centers the form on the screen. Using the
ezSizeMode
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'Input: None
'Output: None
'References:
'      - clFormWindow
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        .Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

End Sub

```

### **FORMCLASS-1-0-0-3-PopUpFrm-MishapDescription**

```
Option Compare Database
Option Explicit
#####
      FORM DESCRIPTION
#####
'Class Name: 1-0-0-3-PopUpFrm-MishapDescription
'
'Author: Pat Flanders & Scott Tufts
'
This class is
'
References:
'   - clFormWindow
'   - ez_SizingFunctions
'   - GlobalDeclarations
'
#####

*****
'
      FUNCTIONS
*****

=====
Function/Sub Name: cmdDone_Click()
'
Description: Closes the form.
'
Input: None
'
Output: None
'
References: None
'
=====
Private Sub cmdDone_Click()
    DoCmd.Close acForm, "1-0-0-3-PopUpFrm-MishapDescription"
End Sub

=====
Function/Sub Name: Form_Activate()
'
Description: Update the menu bar.
'
Input: None
'
Output: None
'
References: None
'
=====
Private Sub Form_Activate()
    Application.CommandBars("mnuOther").Visible = True
End Sub

=====
Function/Sub Name: Form_Deactivate()
'
Description: Updates the menu bar.
'
Input: None
'
Output: None
'
References: None
'
=====
Private Sub Form_Deactivate()
    Application.CommandBars("mnuOther").Visible = False
End Sub

=====
Function/Sub Name: Form_Load()
'
Description: Dynamically resizes the form to the users
screen
resolution and then centers it.
'
Input: None
'
Output: None
'
References:
'   - ezSizeForm
'
=====
Private Sub Form_Load()
    ezSizeForm Me, -1
    MoveToCenter "1-0-0-3-PopUpFrm-MishapDescription"
End Sub

=====
Function/Sub Name: Form_Open()
'
Description: Updates the menu bar and sets shows the value
of the
description for the mishap stored in the
GlobalDeclarations.gStrDescription
global variable.
'
Input: None
'
Output: None
'
References:
'   - GlobalDeclarations
'
=====
Private Sub Form_Open(Cancel As Integer)
    Application.CommandBars("mnuOther").Visible = True
    Me.txtDescription = GlobalDeclarations.gStrDescription
End Sub

=====
Function/Sub Name: MoveToCenter()
'
```

```
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'
'      - clFormWindow
```

```
'
=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        '.Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

End Sub
```

### **FORMCLASS-1-0-0-4-subfrm-Factors**

```
Option Compare Database
Option Explicit
'#####
'          FORM DESCRIPTION
'#####
'Class Name: 1-0-0-4-subfrm-Factors
'
'Author: Pat Flanders & Scott Tufts
'
'This class is used in a form/subform relationship with the
'1-0-0-2-frm-EditMishap form to display, add, and delete
factors
'to a mishap.
'
'References:
'   - 1-0-0-2-frm-EditMishap
'   - clFormWindow
'   - ez_SizingFunctions
'   - GlobalDeclarations
'
'#####

*****
'          FUNCTIONS
*****

'=====
'Function/Sub Name: cmdAddFactor_Click()
'
'Description: Adds a blank factor to the mishap indicated by
the
'GlobalDeclarations.gLngMishapToGet global variable.
'
'Input: None
'
'Output: None
'
'References:
'   - GlobalDeclarations
'
'=====
=
Private Sub cmdAddFactor_Click()

    On Error GoTo Err_cmdAddFactor_Click

    DoCmd.SetWarnings (False) 'Turn off warning messages
    Me.AllowAdditions = True 'Toggle the form to allow
addition of records

    DoCmd.GoToRecord , , acNewRec 'Create a new record
    Me.txtMishapID.Value =
GlobalDeclarations.gLngMishapToGet 'Set the value of the
Mishap
    Me.txtFactorSummary.Value = "Please enter a short
summary description of the Factor."

    Me.cbo3rdLevelCode.Value = "UNK"
    DoCmd.DoMenuItem acFormBar, acRecordsMenu,
acSaveRecord, , acMenuVer70 'Save the record
    Me.AllowAdditions = False 'Toggle back to not allow
addition of records
    Me.Refresh 'Refresh so the user can see the changes
    Me.Recordset.MoveLast 'Move to the record just created
    DoCmd.SetWarnings (True)

Exit_cmdAddFactor_Click:
Exit Sub

Err_cmdAddFactor_Click:

    MsgBox ERR.Description
    Resume Exit_cmdAddFactor_Click

End Sub

'=====
'Function/Sub Name: cmdDelFactor_Click()
'
'Description: Deletes the factor with the current focus.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub cmdDelFactor_Click()
On Error GoTo Err_cmdDelFactor_Click

    'Uncomment this Code to add constraints to ensure at -least
1 Factor per Mishap
    If Me.txtRecordCount.Value = 1 Then
        DoCmd.Beep
        MsgBox "Every project must have at least one Factor.
You can't delete the last Factor, but you can modify it.",
vbOKOnly + vbExclamation, "You Must Have One Factor"
        GoTo Exit_cmdDelFactor_Click
    End If

    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, ,
acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, ,
acMenuVer70

Exit_cmdDelFactor_Click:
Exit Sub

Err_cmdDelFactor_Click:
MsgBox ERR.Description
Resume Exit_cmdDelFactor_Click

End Sub
```

## **FORMCLASS-1-0-0-5-fm-AddMishap**

Option Compare Database

Option Explicit

#####  
' FORM DESCRIPTION  
#####

'Class Name: 1-0-0-5-fm-AddMishap  
,

'Author: Pat Flanders & Scott Tuft s  
,

'This class is a wizard used to add Mishaps to the database.

The

'illusion of many forms is created using a TAB control on the  
form

'and setting the "tab sytle" property to "None". THIS IS  
IMPORTANT.

'The only way to edit the other pages of the tab control is to  
'set the tab property to "Tabs" when the form is in design  
view

'and then change it back to "None" when finished. If you  
don't

'do this, you cannot edit any of the pages of the wizard except  
'the first one.  
,

'After a mishap is added, the 1-0-0-2-fm-EditMishap form is  
'opened with the newly added Mishap selected for editing.

This

'allows the user to immediately add Factors without having to  
'go back to the main menu.  
,

'References:

' - 1-0-0-7-PopUpFrm-CodeMaintenance  
' - 1-0-0-2-fm-EditMishap  
' - clFormWindow  
' - ez\_SizingFunctions  
' - GlobalDeclarations  
,

#####

\*\*\*\*\*  
' FUNCTIONS  
\*\*\*\*\*

=====

'Function/Sub Name: cmdBack\_Click()

'Description: Switches form focus back one tab in the tab  
view  
'control.

'Input: None  
,

'Output: None  
,

'References: None  
,

=====

Private Sub cmdBack\_Click()

Me.cmdFinish.Enabled = False  
DoCmd.GoToControl "Page1"

Me.cmdBack.Enabled = False

Me.cmdNext.Enabled = True

End Sub

=====

'Function/Sub Name: cmdCancel\_Click()

'Description: Closes the form undoing changes.  
,

'Input: None  
,

'Output: None  
,

'References: None  
,

=====

Private Sub cmdCancel\_Click()

On Error GoTo Err\_CmdCancel\_Click

DoCmd.DoMenuItem acFormBar, acEditMenu, acUndo, ,  
acMenuVer70  
Forms![1-0-0-0-fm-SelectMishap].Visible = True  
DoCmd.Close

Exit\_CmdCancel\_Click:  
Exit Sub

Err\_CmdCancel\_Click:  
DoCmd.Close

End Sub

=====

'Function/Sub Name: cmdBack\_Click()

'Description: Switches form focus forward one tab in the tab  
view  
'control.

'Input: None  
,

'Output: None  
,

'References: None  
,

=====

Private Sub cmdNext\_Click()

Me.cmdFinish.Enabled = True  
Me.cmdBack.Enabled = True  
DoCmd.GoToControl "Page2"  
Me.cmdNext.Enabled = False

End Sub

=====

'Function/Sub Name: cmdFinish\_Click()

```

'
'Description: Adds the mishap to the database and opens the
edit
'form so that the user can add factors.
'
'Input: None
'
'Output: None
'
'References:
'   - 1-0-0-2-frm-EditMishap
'
=====
Private Sub cmdFinish_Click()

    On Error GoTo startError

    'Set the database type from the global variable
    Me.txtDatabaseType.Value =
GlobalDeclarations.gStrTypeDB

    'If there is a problem, make it "M" as a default
    If Me.txtDatabaseType.Value <> "M" Or
Me.txtDatabaseType.Value <> "C" Then
        Me.txtDatabaseType.Value = "M"
    End If

    'Save the record
    DoCmd.DoMenuItem acFormBar, acRecordsMenu,
acSaveRecord, , acMenuVer70

    'Note: There was no way to capture the new MishapID
    created by the line
    'above, so when the Edit form is opened, it just goes to the
    last mishap.

    Me.Visible = False 'Make the form invisible so there is no
    screen flickering
    Me.Refresh 'Refresh so the changes takes

    'Open the new Project in the Edit Form so the user can add
    factors
    GlobalDeclarations.gBlnAddAMishap = True
    GlobalDeclarations.gFormNeedsRefresh = True

    DoCmd.OpenForm "1-0-0-2-frm-EditMishap"

exitSub:
    Exit Sub

startError:
    DoCmd.Beep
    MsgBox "You have left at least one field in this wizard
    blank. All entries are mandatory. Please go back and input
    data for all fields.", vbOKOnly, "All Entries Are Mandatory"
    Resume exitSub

End Sub

=====
'Function/Sub Name: cmdCodeMaintenance_Click()
'
'Description: Opens the code maintenance form.
'
'Input: None
'
'Output: None

```

```

'
'References:
'   - 1-0-0-7-PopUpFrm-CodeMaintenance
'
=====
Private Sub cmdCodeMaintenance_Click()
    DoCmd.OpenForm "1-0-0-7-PopUpFrm-
CodeMaintenance"
End Sub

=====
'Function/Sub Name: Form_Activate()
'
'Description: Update the menu bar.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Form_Activate()
    Application.CommandBars("mnuOther").Visible = True
End Sub

=====
'Function/Sub Name: Form_Deactivate()
'
'Description: Updates the menu bar.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Form_Deactivate()
    Application.CommandBars("mnuOther").Visible = False
End Sub

=====
'Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'
'References:
'   - ezSizeForm
'
=====
Private Sub Form_Load()
    ezSizeForm Me, -1
    MoveToCenter "1-0-0-5-frm-AddMishap"
End Sub

Private Sub Form_Open(Cancel As Integer)
    Application.CommandBars("mnuOther").Visible = True
End Sub

```



```

Private Sub txtDate_GotFocus()
    'Format the date in the textbox so the time doesn't appear
    Me.txtDate = Format([txtDate], "Medium Date")
End Sub

'=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeMode
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None

```

```

'
'Output: None
'
'References:
'    - clFormWindow
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        '.Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing
End Sub

```

## **FORMCLASS-1-0-0-6-PopUpFrm-AdministratorLogon**

Option Compare Database  
Option Explicit

'Reusable variable for opening a connection  
Dim conn As New ADODB.Connection

'Reusable variable for recordset operations  
Dim rst As New ADODB.Recordset

#####  
' FORM DESCRIPTION  
#####  
'Class Name: 1-0-0-6-PopUpFrm-AdministratorLogon  
,

'Author: Pat Flanders & Scott Tufts  
,

'This class controls access to the Administrator functions of  
the  
'database. It provides a user logon and compares the User ID  
'and password that are input to values retrieved from a hidden  
'password table in the investigate.mdb database. If the User ID  
'and password match, then the 1-0-0-6-Frm-SelectMishap  
form  
'is opened.  
,

'NOTE: The investigate.mdb database is not encrypted and  
should  
'be replaced with more secure means of validation such as a  
key  
'server in future versions of this program.  
,

'References:  
' - Investigate.mdb  
' - clFormWindow  
' - ez\_SizingFunctions  
' - GlobalDeclarations  
,

#####

\*\*\*\*\*  
' FUNCTIONS  
\*\*\*\*\*

=====

'Function/Sub Name: cboUser\_AfterUpdate()

'Description: Populates the User combo box.  
,

'Input: None  
,

'Output: None  
,

'References: None  
,

=====

Private Sub cboUser\_AfterUpdate()

    rst.MoveFirst  
    Do Until rst.EOF  
        If rst!UID = Me.cboUser.Value Then

        Exit Sub  
    End If  
    rst.MoveNext  
    Loop  
    Me.txtPassword.Value = ""

End Sub

=====

'Function/Sub Name: cmdCancel\_Click()

'Description: Closes the form undoing changes.  
,

'Input: None  
,

'Output: None  
,

'References: None  
,

=====

Private Sub cmdCancel\_Click()

    DoCmd.Close acForm, "1-0-0-6-PopUpFrm-  
AdministratorLogon"

End Sub

=====

'Function/Sub Name: cmdOK\_Click()

'Description: Calls the function to check the password/User  
combination'  
'If successful, sets the global flag so the user doesn't have to  
keep  
'logging on every time he/she wants to access administrative  
functions.  
,

'Input: None  
,

'Output: None  
,

'References:  
' - Globaldeclarations  
' - chkPassword()  
,

=====

Private Sub cmdOK\_Click()

    'Check to see if the user left the password box blank  
    If Trim(Me.txtPassword.Value) = "" Then Exit Sub

    'Call the check password subroutine. If successful, set the  
global  
    'flag so the user doesn't have to keep logging on every time  
he/she  
    'wants to access administrative functions.  
    If chkPassword(Me.cboUser.Value,  
Me.txtPassword.Value) = True Then  
        MsgBox "Login successful." & Chr(13) & Chr(13) &  
"You will not be required to log in again this session.",  
vbInformation + vbOKOnly, "Login Successful"  
        Me.Visible = False

```

        DoEvents
        DoCmd.OpenForm "1-0-0-0-frm-SelectMishap"
    Else
        MsgBox "Invalid password.", vbExclamation +
vbOKOnly, "Login Denied"
    End If

```

```

        DoCmd.Close acForm, "1-0-0-6-PopUpFrm-
AdministratorLogon"

```

```

End Sub

```

```

'=====
'Function/Sub Name: Form_Close()
'

```

```

'Description: Closes the form.
'

```

```

'Input: None
'

```

```

'Output: None
'

```

```

'References: None
'

```

```

'=====
Private Sub Form_Close()

```

```

    'Clean up
    rst.Close
    conn.Close

```

```

End Sub

```

```

'=====
'Function/Sub Name: Form_Load()
'

```

```

'Description: Dynamically resizes the form to the users
screen
resolution and then centers it.
'

```

```

'Input: None
'

```

```

'Output: None
'

```

```

'References:
'    - ezSizeForm
'

```

```

'=====
Private Sub Form_Load()

```

```

    'Dynamically resize the form based on screen resolution.
    ezSizeForm Me, -1
    MoveToCenter "1-0-0-6-PopUpFrm-
AdministratorLogon"

```

```

End Sub

```

```

'=====
'Function/Sub Name: Form_Open()
'

```

```

'Description: Get the user list and passwords from the
Investigate.mdb
file and populate the user combobox with entries.
'

```

```

'Input: None
'

```

```

'Output: None
'

```

```

'References:
'    - Investigate.mdb
'

```

```

'=====
Private Sub Form_Open(Cancel As Integer)

```

```

    'Set the provider name
    conn.Provider = "Microsoft.Jet.OLEDB.4.0"

```

```

    'Open a connection to the data
    conn.Open GlobalDeclarations.gStrAppPath &
"Investigate.mdb"

```

```

    'Open a recordset with a keyset cursor
    rst.Open "SELECT * FROM tblPasswordFile", conn,
adOpenDynamic, adLockOptimistic, adCmdText

```

```

    Dim sValueList As String
    'Walk the recordset creating a list for the combobox to use.
    Do Until rst.EOF
        sValueList = rst!UID & ";" & sValueList
        rst.MoveNext
    Loop

```

```

    'Populate the combobox.
    Me.cboUser.RowSource = sValueList
    rst.MoveFirst
    Me.cboUser.Value = rst!UID
    Me.txtPassword.Value = ""

```

```

End Sub

```

```

'=====
'Function/Sub Name: chkPassword()
'

```

```

'Description: Checks the User/Password combination for
validity.
'

```

```

'Input:
'    - strUID      User Name as String
'    - strPWD      User password as string
'

```

```

'Output: Success or failure.
'

```

```

'References:
'    - Investigate.mdb
'

```

```

'=====
Private Function chkPassword(strUID As String, strPWD As
String) As Boolean

```

```

    rst.MoveFirst

```

```

    'Walk the recordset if a userID/Password combination
match is
    'found, return success.
    Do Until rst.EOF

```

```

        If rst!UID = Trim(strUID) And rst!PWD =
Trim(strPWD) Then
            GlobalDeclarations.gBlnAdministrator = True
            chkPassword = True
            Exit Do
        Else
            GlobalDeclarations.gBlnAdministrator = False

```

```

        chkPassword = False
    End If

```

```

    rst.MoveNext

```

```

    Loop

```

```

End Function

```

```

'=====
'Function/Sub Name:  MoveToCenter()
'
'Description:  Centers the form on the screen.  Using the
ezSizeForm
'class breaks Access's built -in autocenter function, so this
'method is needed to fix it.  Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'

```

```

'Input: None
'

```

```

'Output: None
'

```

```

'References:
'
'    - clFormWindow
'

```

```

'=====
Public Sub MoveToCenter(ByVal strFormName As String)

```

```

    Dim fwForm As New clFormWindow

```

```

    With fwForm

```

```

        .hwnd = Forms(strFormName).hwnd

```

```

        '.Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)

```

```

        .Left = (.Parent.Width - .Width) / 2

```

```

    End With

```

```

    Set fwForm = Nothing

```

```

End Sub

```

## **FORMCLASS-1-0-0-7-PopUpFrm-CodeMaintenance**

```
Option Compare Database
Option Explicit
'#####
'      FORM DESCRIPTION
'#####
'Class Name: 1-0-0-7-PopUpFrm-CodeMaintenance
'
'Author: Pat Flanders & Scott Tufts
'
'Allows an Administrator to add codes directly to the database
code
'lookup tables.
'
'References:
'      - tblAircraft
'      - tblMishapClass
'      - tblMishapLocation
'      - tblOrganization
'      - tblmishaptype
'
'#####

'*****
'      FUNCTIONS
'*****
```

```
'=====
'Function/Sub Name: cmdClose_Click()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub cmdClose_Click()
    DoCmd.Close acForm, "1-0-0-7-PopUpFrm-CodeMaintenance"
End Sub
```

```
'=====
'Function/Sub Name: cmdOK_Click()
'
'Description: Opens the appropriate table for direct editing
based
'on the radio button selection in the frame.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub cmdOK_Click()
```

```
    If Me.Frame6 = 1 Then
        DoCmd.OpenTable "dbo.tblAircraft", acViewNormal,
acEdit
    End If

    If Me.Frame6 = 2 Then
        DoCmd.OpenTable "dbo.tblMishapClass",
acViewNormal, acEdit
    End If

    If Me.Frame6 = 3 Then
        DoCmd.OpenTable "dbo.tblMishapLocation",
acViewNormal, acEdit
    End If

    If Me.Frame6 = 4 Then
        DoCmd.OpenTable "dbo.tblOrganization",
acViewNormal, acEdit
    End If

    If Me.Frame6 = 5 Then
        DoCmd.OpenTable "dbo.tblmishaptype",
acViewNormal, acEdit
    End If

End Sub
```

```
'=====
'Function/Sub Name: Form_Activate()
'
'Description: Update the menu bar.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Activate()
    'Change the menus when the form is activated (uncovered)
    Application.CommandBars("mnuOther").Visible = True
    Application.CommandBars("FindSortExport").Visible =
False
End Sub
```

```
'=====
'Function/Sub Name: Form_Close()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Close()
    'Change the menus when the form is activated (uncovered)
    Application.CommandBars("mnuOther").Visible = True
```

```

Application.CommandBars("FindSortExport").Visible =
False
End Sub

```

```

'=====
'Function/Sub Name: Form_Deactivate()
'

```

```

'Description: Updates the menu bar.
'

```

```

'Input: None
'

```

```

'Output: None
'

```

```

'References: None
'

```

```

'=====
Private Sub Form_Deactivate()

```

```

    'Change the menus when the form is covered up
    (deactivated)
    Application.CommandBars("mnuOther").Visible = False
    Application.CommandBars("FindSortExport").Visible =
    True

```

```

End Sub

```

```

'=====
'Function/Sub Name: Form_Load()
'

```

```

'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'

```

```

'Input: None
'

```

```

'Output: None
'

```

```

'References:
'
    - ezSizeForm

```

```

'=====
Private Sub Form_Load()

```

```

    ezSizeForm Me, -1
    MoveToCenter "1-0-0-7-PopUpFrm-CodeMaintenance"
End Sub

```

```

'=====
'Function/Sub Name: MoveToCenter()
'

```

```

'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'

```

```

'Input: None
'

```

```

'Output: None
'

```

```

'References:
'
    - clFormWindow

```

```

'=====
Public Sub MoveToCenter(ByVal strFormName As String)

```

```

    Dim fwForm As New clFormWindow

```

```

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        '.Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

```

```

End Sub

```

## **FORMCLASS-1-0-0-8-PopUpFrm-PasswordMaintenance**

Option Compare Database  
Option Explicit

'Reusable variable for opening a connection  
Dim conn As New ADODB.Connection

'Reusable variable for recordset operations  
Dim rst As New ADODB.Recordset

'Flag for differentiating new entries from edits  
Dim bNewOrEdit As Boolean

#####  
' FORM DESCRIPTION  
#####  
'Class Name: 1-0-0-8-PopUpFrm-PasswordMaint

'Author: Pat Flanders & Scott Tufts

'This class controls access to the Administrator functions of  
the  
'of the password table in the Investigate.mdb database. The  
table  
'is HIDDEN and cannot be viewed directly. This class  
allows addition,  
'deletion, and editing of passwords and user IDs in THAT  
database.

'NOTE: The Investigate.mdb database is not encrypted and  
should  
'be replaced with more secure means of validation such as a  
key  
'server in future versions of this program.

References:  
' - Investigate.mdb  
' - clFormWindow  
' - ez\_SizingFunctions  
' - GlobalDeclarations

#####

\*\*\*\*\*  
' FUNCTIONS  
\*\*\*\*\*

=====

'Function/Sub Name: cboUser\_AfterUpdate()

'Description: Populates the User combo box.

'Input: None

'Output: None

'References: None

=====

Private Sub cboUser\_AfterUpdate()

rst.MoveFirst  
Do Until rst.EOF  
If rst!UID = Me.cboUser.Value Then  
Me.txtClearPWD.Value = rst!PWD  
Exit Sub  
End If  
rst.MoveNext  
Loop

End Sub

=====

'Function/Sub Name: cmdCancel\_Click()

'Description: Closes the form undoing changes.

'Input: None

'Output: None

'References: None

=====

Private Sub cmdCancel\_Click()  
  
rst.CancelUpdate  
DoCmd.Close acForm, "1-0-0-8-PopUpFrm-  
PasswordMaint"

End Sub

=====

'Function/Sub Name: cmdDelete\_Click()  
'Description: Deletes the selected user from the password  
table.

'Input: None

'Output: None

'References: None

=====

Private Sub cmdDelete\_Click()  
  
Dim response As Variant  
response = MsgBox("Ary you sure you want to delete the  
HFACS Administration account for " & Me.cboUser.Value  
& "?", vbYesNo + vbQuestion, "Delete Admin Account")  
If response = vbYes Then  
rst.Delete  
DoCmd.Close acForm, "1-0-0-8-PopUpFrm-  
PasswordMaint"

End Sub

=====

'Function/Sub Name: cmdNew\_Click()

```

'Description: Adds a new user to the password table.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub cmdNew_Click()

    Me.cboUser.Visible = False
    Me.txtUser.Visible = True
    Me.txtClearPWD = ""
    Me.txtPassword.Value = ""
    Me.txtConfirm.Value = ""
    Me.cmdDelete.Enabled = False
    rst.AddNew
    bNewOrEdit = True
    Me.txtUser.SetFocus

End Sub

'=====
'Function/Sub Name: cmdSave_Click()
'
'Description: Validates entries and saves changes.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub cmdSave_Click()

    On Error GoTo startError

    'Make sure password and password confirmation match.
    If Trim(Me.txtPassword.Value) =
Trim(Me.txtConfirm.Value) Then

        Select Case bNewOrEdit

            Case True 'This is a new entry, so make sure both User
& Password are specified.

                If Trim(Me.txtUser.Value) = "" Or
Trim(Me.txtPassword.Value) = "" Then
                    MsgBox "You can't leave the USER or
PASSWORD fields blank.", vbOKOnly + vbExclamation,
"Missing Data"
                    Exit Sub
                End If

                DoCmd.SetWarnings (False)
                rst!UID = Me.txtUser.Value
                rst!PWD = Me.txtPassword.Value
                rst.Update
                DoCmd.SetWarnings (True)

            Case False 'This is an edit, so make sure the password is
specified.

                If Trim(Me.txtPassword.Value) = "" Then

```

```

                    MsgBox "You can't leave the PASSWORD field
blank.", vbOKOnly + vbExclamation, "Missing Data"
                    Exit Sub
                End If

                'Update the database with the changes.
                DoCmd.SetWarnings (False)
                rst!UID = Me.cboUser.Value
                rst!PWD = Me.txtPassword.Value
                rst.Update
                DoCmd.SetWarnings (True)

            End Select

            DoCmd.Close acForm, "1-0-0-8-PopUpFrm-
PasswordMaint"

        Else
            MsgBox "Your new password and confirmation entries
do not match.", vbOKOnly + vbExclamation, "Passwords
Don't Match"
            Me.txtPassword.Value = ""
            Me.txtConfirm.Value = ""
        End If

    exitSub:

        Exit Sub

    startError:
        MsgBox ERR.Description & "Number: " & ERR.Number
        GoTo exitSub

End Sub

'=====
'Function/Sub Name: Form_Close()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Close()

    'Clean up
    rst.Close
    conn.Close

End Sub

'=====
'Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'

```



References:

' - ezSizeMode

```
Private Sub Form_Load()  
    ezSizeMode Me, -1  
    MoveToCenter "1-0-0-8-PopUpFrm-PasswordMaint"  
End Sub
```

Function/Sub Name: Form\_Open()

Description: Get the user list and passwords from the Investigate.mdb file and populate the user combobox with entries.

Input: None

Output: None

References:

' - Investigate.mdb

Private Sub Form\_Open(Cancel As Integer)

'Set the provider name  
conn.Provider = "Microsoft.Jet.OLEDB.4.0"

'Open a connection to the data  
conn.Open GlobalDeclarations.gStrAppPath &  
"Investigate.mdb"

'Open a recordset with a keyset cursor  
rst.Open "SELECT \* FROM tblPasswordFile", conn,  
adOpenDynamic, adLockOptimistic, adCmdText

Dim sValueList As String  
'Walk the recordset  
Do Until rst.EOF  
 sValueList = rst!UID & ";" & sValueList  
 rst.MoveNext  
Loop

Me.cboUser.RowSource = sValueList

rst.MoveFirst  
Me.cboUser.Value = rst!UID  
Me.txtClearPWD.Value = rst!PWD  
Me.txtPassword.Value = ""  
Me.txtConfirm.Value = ""

End Sub

Function/Sub Name: txtUser\_GotFocus()

Description: Disable the "New" button once it has been clicked.

Input: None

Output: None

References: None

Private Sub txtUser\_GotFocus()

Me.cmdNew.Enabled = False

End Sub

Function/Sub Name: MoveToCenter()

Description: Centers the form on the screen. Using the ezSizeMode class breaks Access's built-in autocenter function, so this method is needed to fix it. Each form gets its own version of this function so that minor adjustments can be made on a form by basis.

Input: None

Output: None

References:

' - clFormWindow

Public Sub MoveToCenter(ByVal strFormName As String)

Dim fwForm As New clFormWindow

With fwForm  
 .hwnd = Forms(strFormName).hwnd  
 .Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) \*  
0.6)  
 .Left = (.Parent.Width - .Width) / 2  
End With  
Set fwForm = Nothing

End Sub

## **FORMCLASS-2-0-1-0-frm-QueryMenu**

Option Compare Database  
Option Explicit

```
#####  
' FORM DESCRIPTION  
#####  
'Class Name: 2-0-1-0-frm-QueryMenu  
'  
'Author: Pat Flanders & Scott Tufts  
'  
'This class is the form for selecting the type of query to run.  
'It has no special functionality or recordsource.  
'  
'References:  
' - clFormWindow  
' - ez_SizingFunctions  
' - GlobalDeclarations  
' - 2-0-1-1-frm-ExpertQueryForm  
' - 2-0-2-1-frm-Summary  
'
```

#####

```
*****  
' FUNCTIONS  
*****
```

=====

```
'Function/Sub Name: cmdCloseQueryMenu_Click()
```

'Description: Closes the form.

'Input: None

'Output: None

'References: None

=====

```
Private Sub cmdCloseQueryMenu_Click()  
DoCmd.Close acForm, "2-0-1-0-frm-QueryMenu"  
End Sub
```

=====

```
'Function/Sub Name:
```

```
' - cmdCloseQueryMenu_MouseMove()  
' - cmdExpertQuery_MouseMove()  
' - cmdHFACS_MESummary_MouseMove()  
'
```

'Description: The following 3 functions update textcolor on the command buttons in response to mouse over events.

'Input: None

'Output: None

'References: None

=====

```
Private Sub cmdCloseQueryMenu_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)  
' Make the button text blue when it gets the focus  
Me.cmdExpertQuery.ForeColor = QBColor(0)  
Me.cmdHFACS_MESummary.ForeColor = QBColor(0)  
Me.cmdCloseQueryMenu.ForeColor = QBColor(9)  
End Sub
```

```
Private Sub cmdExpertQuery_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)  
' Make the button text blue when it gets the focus  
Me.cmdExpertQuery.ForeColor = QBColor(9)  
Me.cmdHFACS_MESummary.ForeColor = QBColor(0)  
Me.cmdCloseQueryMenu.ForeColor = QBColor(0)  
End Sub
```

```
Private Sub cmdHFACS_MESummary_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)  
' Make the button text blue when it gets the focus  
Me.cmdExpertQuery.ForeColor = QBColor(0)  
Me.cmdHFACS_MESummary.ForeColor = QBColor(9)  
Me.cmdCloseQueryMenu.ForeColor = QBColor(0)  
End Sub
```

=====

```
'Function/Sub Name: Form_Close()
```

'Description: Closes the form.

'Input: None

'Output: None

'References: None

=====

```
Private Sub Form_Close()
```

```
Application.CommandBars("mnuOther").Visible = False  
Application.CommandBars("mnuProgramMain").Visible = True  
Forms![MainMenu].Visible = True
```

End Sub

=====

```
'Function/Sub Name: Form_Activate()
```

'Description: Update the menu bar.

'Input: None

'Output: None

'References: None

=====

```
Private Sub Form_Activate()  
Application.CommandBars("mnuOther").Visible = True  
End Sub
```

```

=====
'Function/Sub Name: Form_Deactivate()
'
'Description: Updates the menu bar.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Form_Deactivate()
    Application.CommandBars("mnuOther").Visible = False
End Sub

=====
'Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'
'References:
'    - ezSizeForm
'
=====
Private Sub Form_Load()

    ezSizeForm Me, -1
    MoveToCenter "2-0-1-0-frm-QueryMenu"

End Sub

=====
'Function/Sub Name: Form_Open
'
'Description: Updates the menu bar and sets the focus to the
first
'command button, setting its color to blue.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Form_Open(Cancel As Integer)

    Forms![MainMenu].Visible = False
    Application.CommandBars("mnuOther").Visible = True

    Me.cmdCloseQueryMenu.SetFocus

    ' Make the button text blue when it gets the focus
    Me.cmdExpertQuery.ForeColor = QBColor(0)
    Me.cmdHFACS_MESummary.ForeColor = QBColor(0)
    Me.cmdCloseQueryMenu.ForeColor = QBColor(0)

```

End Sub

```

=====
'Function/Sub Name: cmdHFACS_MESummary_Click()
'
'Description: Opens the 2-0-2-1-frm-Summary form.
'
'Input: None
'
'Output: None
'
'References:
'    - 7-0-0-1-PopUpFrm-waitProgressBar
'
=====
Private Sub cmdHFACS_MESummary_Click()

    DoCmd.OpenForm "7-0-0-1-PopUpFrm-
waitProgressBar", acNormal, "", "", acReadOnly, acNormal
    DoCmd.RepaintObject acForm, "7-0-0-1-PopUpFrm-
waitProgressBar"
    DoCmd.OpenForm "2-0-2-1-frm-Summary"
    DoCmd.Close acForm, "7-0-0-1-PopUpFrm-
waitProgressBar"

End Sub

=====
'Function/Sub Name: cmdExpertQuery_Click()
'
'Description: Opens the 2-0-1-1-frm-ExpertQueryForm
form.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub cmdExpertQuery_Click()

    DoCmd.OpenForm "2-0-1-1-frm-ExpertQueryForm"

End Sub

=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'    - clFormWindow

```

```

'
=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd

        '.Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

End Sub

```

## **FORMCLASS-2-0-1-1-frm-ExpertQueryForm**

Option Compare Database  
Option Explicit

'Reusable variable for creating combobox value lists  
Dim tempvaluelist As String

#####  
' FORM DESCRIPTION  
#####  
'Class Name: 2-0-1-1-frm-ExpertQueryForm  
,

'Author: Pat Flanders & Scott Tufts  
,

'This form allows the user to choose multiple criteria from a series of  
'combo boxes and then query the database to open the  
'2-0-1-2-frm-ViewMishaps form and display the mishaps and factors.  
,

'When the form opens, it populates the combo boxes by running UNION  
'queries to build the recordsets needed to serve as control sources.  
'This is necessary to add the "<All>" choice. The only exception  
'is the "Year" combo box. It uses a string manipulation function  
'called populateComboBoxWithAll() to build a value list. This  
'is necessary because the UNION method will only work with non-integer  
'data types. The problem with the populateComboBoxWithAll()  
'method is that it is limited in size to about 50 two dimensional  
'entries. In addition, commas and semi-colons create problems  
'and must be removed from the string during build.  
,

'Finally, when the user clicks "View", code is executed that builds  
'the input string for stored procedure 2-0-1-1-flanCountflanFilteredMishaps  
'which is the recordsource for the 2-0-1-2-frm-ViewMishaps form.  
'this input string is then passed to the "view" form via a global  
'variable and the viewMishaps form is opened.  
,

'References:  
' - clFormWindow  
' - ez\_SizingFunctions  
' - GlobalDeclarations  
' - 2-0-1-2-frm-ViewMishaps  
,

#####

\*\*\*\*\*  
' FUNCTIONS  
\*\*\*\*\*

=====

'Function/Sub Name: Form\_Activate()

'Description: Update the menu bar.

'Input: None

'Output: None

'References: None  
,

=====

Private Sub Form\_Activate()

Application.CommandBars("mnuOther").Visible = True

End Sub

=====

'Function/Sub Name: Form\_Deactivate()

'Description: Updates the menu bar.

'Input: None

'Output: None

'References: None  
,

=====

Private Sub Form\_Deactivate()

Application.CommandBars("mnuOther").Visible = False

End Sub

=====

'Function/Sub Name: Form\_Close()

'Description: Updates the menu bar.

'Input: None

'Output: None

'References: None  
,

=====

Private Sub Form\_Close()

Application.CommandBars("mnuOther").Visible = False

End Sub

=====

'Function/Sub Name: cmdBack\_Click()

'Description: Closes the form.

'Input: None

'Output: None

'References: None  
,

=====

```
Private Sub cmdBack_Click()
    DoCmd.Close acForm, "2-0-1-1-fm-ExpertQueryForm"
End Sub
```

```
=====
'Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'
'References:
'
'    - ezSizeForm
'
=====
Private Sub Form_Load()
```

```
    ezSizeForm Me, -1
    MoveToCenter "2-0-1-1-fm-ExpertQueryForm"

End Sub
```

```
=====
'Function/Sub Name: Form_Open()
'
'Description: Populates combo boxes. In order to allow the
combo
'boxes to offer <All> as a choice, 2 methods are
'needed -- one for integers and another for strings.

'The populateComboBoxWillAll() subroutine is used for
integers (like the Mishap
'Year), while stored procedures are used for strings.

'Important to note that the populateComboBoxWillAll() will
not work for
'creating strings of more than about 50 entries because the
combo box
'rejects them as too, long. Stored procedures, however, do
not suffer
'from this limitation.
'
```

```
'Input: None
'
'Output: None
'
'References:
'
'    - populateComboBoxWithAll()
'
=====
Private Sub Form_Open(Cancel As Integer)
```

```
    Application.CommandBars("mnuOther").Visible = True

    'Populate each combo box
    'Aircraft
    With Me.cboAircraft
        .RowSource = "9-0-0-2-flanLookupAircraftAll"
        .Value = "<All>"
    End With

    'Organization
```

```
With Me.cboOrganization
    .RowSource = "9-0-0-2-flanLookupOrganizationAll"
    .Value = "<All>"
End With
```

```
'Location
With Me.cboLocation
    .RowSource = "9-0-0-2-flanLookupLocationAll"
    .Value = "<All>"
End With
```

```
'Class category
With Me.cboClass
    .RowSource = "9-0-0-2-flanLookupClassAll"
    .Value = "<All>"
End With
```

```
'Type category
With Me.cboType
    .RowSource = "9-0-0-2-flanLookupTypeAll"
    .Value = "<All>"
End With
```

```
'Year (can't use UNION stored procedure to append <All>
because it is of type Integer)
populateComboBoxWithAll "9-0-0-2-
flanModifiedLookupYear", 1
With Me.cboYear
    .RowSourceType = "Value List"
    .RowSource = tempvaluelist
    .Value = "<All>"
End With
```

```
End Sub
```

```
=====
'Function/Sub Name: cmdView_Click()
'
'Description: Builds the input string to pass to the stored
procedure
'to get the correct records. Order of these if statements must
match the SP.
'If <All> was selected, then pass " so that the SP knows the
value is NULL.

'Once the input string is built, a stored procedure is run from
within
'this function to determine if there are actually any records in
the
'database matching the users selections. If no records match,
ane error
'message is displayed. Otherwise the 2-0-1-2-fm-
ViewMishaps form
'is opened.
'
```

```
'Input: None
'
'Output: None
'
'References:
'
'    - 2-0-1-2-fm-ViewMishaps
'    - gStrInputString
'
=====
Private Sub cmdView_Click()
```

```
    On Error GoTo Err_cmdView_Click
```

```

'Reset the global variable
GlobalDeclarations.gStrInputString = ""

'Build the input string to pass to the stored procedure to get
the correct records.
'Order of these if statements must match the SP.
'If <All> was selected, then pass " so that the SP knows the
value is NULL.
If Me.cboAircraft.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString = "@AC
varchar(10)=" & Me.cboAircraft.Value & ""
Else
    GlobalDeclarations.gStrInputString = "@AC
varchar(10)="
End If

If Me.cboType.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Type
varchar(3)=" & Me.cboType.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Type
varchar(3)="
End If

If Me.cboClass.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Class
varchar(1)=" & Me.cboClass.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Class
varchar(1)="
End If

If Me.cboLocation.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Loc
varchar(25)=" & Me.cboLocation.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Loc
varchar(25)="
End If

If Me.cboOrganization.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Svc
varchar(10)=" & Me.cboOrganization.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Svc
varchar(10)="
End If

If Me.cboYear.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Yr int=" &
Me.cboYear.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Yr int="
End If

```

These 3 paramaters are required for the SP to run (because the HFACS summary form uses the same SP), but remain NULL

```

GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @1stLevel
varchar(5)="
GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @2ndLevel
varchar(5)="
GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @3rdLevel
varchar(5)="

```

'Run a stored procedure to determine if there are actually any records in the database matching the users selections.

```

Dim cnn As Connection
Dim oCmd As ADODB.Command
Dim rst As ADODB.Recordset

```

```

Dim objPrmAC As ADODB.Parameter
Dim objPrmSvc As ADODB.Parameter
Dim objPrmType As ADODB.Parameter
Dim objPrmClass As ADODB.Parameter
Dim objPrmLoc As ADODB.Parameter
Dim objPrmYr As ADODB.Parameter

```

```

Set cnn = CurrentProject.Connection
cnn.CursorLocation = adUseClient
Set rst = New ADODB.Recordset
Set oCmd = New ADODB.Command
oCmd.ActiveConnection = cnn
oCmd.CommandText = ""2-0-1-1-
flanCountflanFilteredMishaps""
oCmd.CommandType = adCmdStoredProc

```

'Create parameters for the SP that correspond to the combo boxes.

'They have to be appended in the same order that they appear in the stored procedure.

```

Set objPrmAC = oCmd.CreateParameter("@AC",
adVarChar, adParamInput, 10)
oCmd.Parameters.Append objPrmAC
If Me.cboAircraft.Value <> "<All>" Then
    objPrmAC.Value = Me.cboAircraft.Value
End If

```

```

Set objPrmType = oCmd.CreateParameter("@Type",
adVarChar, adParamInput, 3)
oCmd.Parameters.Append objPrmType
If Me.cboType.Value <> "<All>" Then
    objPrmType.Value = Me.cboType.Value
End If

```

```

Set objPrmClass = oCmd.CreateParameter("@Class",
adVarChar, adParamInput, 1)
oCmd.Parameters.Append objPrmClass
If Me.cboClass.Value <> "<All>" Then
    objPrmClass.Value = Me.cboClass.Value
End If

```

```

Set objPrmLoc = oCmd.CreateParameter("@Loc",
adVarChar, adParamInput, 25)
oCmd.Parameters.Append objPrmLoc
If Me.cboLocation.Value <> "<All>" Then

```

```

        objPrmLoc.Value = Me.cboLocation.Value
    End If

    Set objPrmSvc = oCmd.CreateParameter("@Svc",
adVarChar, adParamInput, 10)
    oCmd.Parameters.Append objPrmSvc
    If Me.cboOrganization.Value <> "<All>" Then
        objPrmSvc.Value = Me.cboOrganization.Value
    End If

    Set objPrmYr = oCmd.CreateParameter("@Yr", adInteger,
adParamInput)
    oCmd.Parameters.Append objPrmYr
    If Me.cboYear.Value <> "<All>" Then
        objPrmYr.Value = Me.cboYear.Value
    End If

    'These 3 paramaters are required for the SP to run (because
the HFACS summary form uses the same SP), but remain
NULL
    Set objPrmSvc = oCmd.CreateParameter("@1stLevel",
adVarChar, adParamInput, 10)
    oCmd.Parameters.Append objPrmSvc
    Set objPrmSvc = oCmd.CreateParameter("@2ndLevel",
adVarChar, adParamInput, 10)
    oCmd.Parameters.Append objPrmSvc
    Set objPrmSvc = oCmd.CreateParameter("3rdLevel",
adVarChar, adParamInput, 10)
    oCmd.Parameters.Append objPrmSvc

    'Run the SP
    Set rst = oCmd.Execute

    'Get the record count
    rst.MoveFirst
    Dim tempRecordCount As Integer
    tempRecordCount = rst!NumRecords

    'Clean up
    rst.Close
    Set oCmd = Nothing
    cnn.Close

    'If there really are records, then open them up, otherwise,
tell
    'the user that no records matched his search criteria.
    If tempRecordCount > 0 Then
        DoCmd.OpenForm "2-0-1-2-fm-ViewMishaps"
    Else
        MsgBox "There are no records that match your search
criteria.", vbOKOnly + vbInformation, "Criteria Too
Restrictive"
    End If

Exit_cmdView_Click:
    Exit Sub

Err_cmdView_Click:
    MsgBox ERR.Description
    Resume Exit_cmdView_Click

End Sub

```

```

'=====
'Function/Sub Name: populateComboBoxWithAll()
'

```

'Description: Makes a connection to the stored procedure passed-in  
'and builds an string that can be used by a combo box to display  
'and "iNumberColToGet" column drop down list. It has to check  
'every record for commas and semi-colons in the data because these  
'two characters are interpreted by the combo box as delimiters,  
'so they must be replaced with some other character (a "-" is what  
'I'm using here).

```

'Input:
'      sNameOfSP      - Name of the Stored Procedure to
get
'                      the records from.
'
'      iNumberColToGet- Number of columns of data to
read
'                      from the Stored Procedure.

```

```

'Output: None
'
'References:
'      - clFormWindow
'

```

```

Private Sub populateComboBoxWithAll(sNameOfSP As
String, iNumberColToGet As Integer)

```

'STEP 1 - Make a connection and get a recordset matching the passed in parameters

```

    Dim cnn As Connection
    Dim oCmd As ADODB.Command
    Dim rst As ADODB.Recordset
    Set cnn = CurrentProject.Connection
    cnn.CursorLocation = adUseClient
    Set rst = New ADODB.Recordset
    Set oCmd = New ADODB.Command
    oCmd.ActiveConnection = cnn
    oCmd.CommandText = "" & sNameOfSP & ""
    oCmd.CommandType = adCmdStoredProc
    Set rst = oCmd.Execute

```

'Make sure tempvalue list is empty before adding to it.  
tempvaluelist = ""

'STEP 2 - Build a string of all the values starting with choice <All>.

```

    Dim i As Integer
    i = 0
    For i = 0 To (iNumberColToGet - 1)
        tempvaluelist = "<All>," & tempvaluelist 'Add <All>
    Next

```

```

    'Now add the real values
    rst.MoveFirst
    Do Until rst.EOF
        Dim k As Integer
        k = 0
        While k < iNumberColToGet

```

'STEP 3 - Replace commas and semicolons with dashes because the mess up the list

```

    Dim astrText As String
    Dim iCount As Integer

```



```

'Check for null fields and only operate on those that
are not null
If IsNull(rst.Fields(k)) Then
    tempvaluelist = tempvaluelist & rst.Fields(k) & ";"
Else
    astrText = Trim(rst.Fields(k))

' Loop through array, replacing commas and
semicolons
For iCount = 1 To Len(astrText)

    If Mid(astrText, iCount, 1) = "," Or
Mid(astrText, iCount, 1) = ";" Then
        ' If array element satisfies wildcard search,
        ' replace it.
        Mid(astrText, iCount, 1) = "-"
    End If
Next
' Join string.
tempvaluelist = tempvaluelist & astrText & ";"
End If
k = k + 1
Wend
rst.MoveNext
Loop

rst.Close
Set rst = Nothing
Set oCmd = Nothing
cnn.Close

End Sub

```

```

=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeMode
'class breaks Access's built -in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'
'          - clFormWindow
'
=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        '.Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

End Sub

```

## **FORMCLASS-2-0-1-2-fm-ViewMishaps**

Option Compare Database  
Option Explicit

'#####  
' FORM DESCRIPTION  
'#####

'Class Name: 2-0-1-2-fm-ViewMishaps

'Author: Pat Flanders & Scott Tufts

'This class is used to view the mishaps with factors. It does  
'NOT allow input, edit, or deletion of data. It is called by  
'both the 2-0-1-1-fm-ExpertQueryForm and the 2-0-2-1-fm-  
'Summary  
'form.

'Because it is called by two different forms, it has the  
'capability  
'to determine which stored procedure to use as a record  
'source  
'based on the value of the  
'GlobalDeclarations.bUseHFACSSummaryQuery  
'global variable.

'References:

' - 2-0-1-2-subFrm-ViewMishaps  
' - clFormWindow  
' - ez\_SizingFunctions  
' - GlobalDeclarations

'#####

\*\*\*\*\*  
' FUNCTIONS  
\*\*\*\*\*

'=====

'Function/Sub Name: cmdCancel\_Click()

'Description: Saves the state of the data (and size of the  
'form)  
'and closes the form.

'Input: None

'Output: None

'References: None

'=====

Private Sub cmdCancel\_Click()

On Error GoTo Err\_CmdCancel\_Click

DoCmd.DoMenuItem acFormBar, acEditMenu, acUndo, ,  
acMenuVer70  
DoCmd.Close

Exit\_CmdCancel\_Click:  
Exit Sub

Err\_CmdCancel\_Click:  
DoCmd.Close

End Sub

'=====

'Function/Sub Name: cmdSave\_Click()

'Description: Closes the form.

'Input: None

'Output: None

'References: None

'=====

Private Sub cmdDone\_Click()

DoCmd.Close acForm, "2-0-1-2-fm-ViewMishaps"

End Sub

'=====

'Function/Sub Name: Form\_Activate()

'Description: Update the menu bar.

'Input: None

'Output: None

'References: None

'=====

Private Sub Form\_Activate()  
Application.CommandBars("mnuOther").Visible = True  
End Sub

'=====

'Function/Sub Name: Form\_Close()

'Description: Resets the flag used to tell the form which  
'stored  
'procedure to use for a record source.

'Input: None

'Output: None

'References: None

'=====

Private Sub Form\_Close()  
GlobalDeclarations.bUseHFACSSummaryQuery = False  
End Sub

'=====

'Function/Sub Name: Form\_Deactivate()

```

'Description: Updates the menu bar.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Form_Deactivate()
    Application.CommandBars("mnuOther").Visible = False
End Sub

'
=====
Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'
'References:
'    - ezSizeForm
'
=====
Private Sub Form_Load()
    ezSizeForm Me, -1
    MoveToCenter "2-0-1-2-frm-ViewMishaps"
End Sub

'
=====
Function/Sub Name: Form_Open()
'
'Description: Determines which stored procedure to use as a
record source
'based on the value of the
GlobalDeclarations.bUseHFACSSummaryQuery
'global variable.
'
'Input: None
'
'Output: None
'
'References:
'    - GlobalDeclarations
'
=====
Private Sub Form_Open(Cancel As Integer)

    Application.CommandBars("mnuOther").Visible = True

    'Determine which stored procedure to use as a record
source.
    If GlobalDeclarations.bUseHFACSSummaryQuery = True
Then
        'This form was called from the 2-0-2-1-frm-Summary
form.
        Me.RecordSource = "dbo.2-0-2-1-
flanSummaryGetRecords"
    Else
        'This form was called from the 2-0-1-1-frm-
ExpertQueryForm form.

```

```

        Me.RecordSource = "dbo.2-0-1-1-
flanFilteredMishapTable"
    End If

    'Set the unique table for the underlying stored procedure
with code
    'becuase it sometimes dissapears when using the visual
property sheet.
    Me.UniqueTable = "tblMishaps"

    'Set the inputparameters for opening the form
    Me.InputParameters = GlobalDeclarations.gStrInputString

End Sub

'
=====
Function/Sub Name: cmdPreview_Click()
'
'Description: Opens a report.
'
'Input: None
'
'Output: None
'
'References:
'    - 1-0-MishapSnapshot-OpenMishaps
'
=====
Private Sub cmdPreview_Click()

    GlobalDeclarations.gLngMishapToGet = Me.txtMishapID

    On Error GoTo startError

    DoCmd.OpenReport "1-1-MishapSnapShot",
acViewPreview

exitSub:

Exit Sub

startError:

    MsgBox "You must have a default printer installed in
order to preview reports.", vbCritical + vbOKOnly, "Can't
Find A Printer"

End Sub

'
=====
Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None

```

```

'
'References:
'      - clFormWindow
'
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        '.Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing
End Sub

```

## **FORMCLASS-2-0-2-1-frm-Summary**

Option Compare Database  
Option Explicit

'Reusable variable for creating combobox value lists  
Dim tempvaluelist As String

'Used to track if combo boxes have been changed, but no  
update has been performed.  
Dim bUpdateNeeded As Boolean

'Variables for storing initial values.  
'Used for tracking if the user actually changed something.  
Dim sStoredAircraft As String  
Dim sStoredType As String  
Dim sStoredClass As String  
Dim sStoredLocation As String  
Dim sStoredOrganization As String  
Dim vStoredYear As Variant  
Dim sStored1stLevel As String  
Dim sStored2ndLevel As String  
Dim sStored3rdLevel As String

#####  
' FORM DESCRIPTION  
#####  
'Class Name: 2-0-2-1-frm-Summary

'Author: Pat Flanders & Scott Tufts

'This class is used to depict the table of factor vs. mishap  
counts  
'and percentages. It allows the user to select criteria from  
combo  
'boxes and fills then calculates the values for the table when  
the  
'user clicks update.

'When the form opens, it populates the combo boxes by  
running UNION  
'queries to build the recordsets needed to serve as control  
sources.  
'This is necessary to add the "<All>" choice. The only  
exception  
'is the "Year" combo box. It uses a string manipulation  
function  
'called populateComboBoxWithAll() to build a value list.  
This  
'is necessary because the UNION method will only work  
with non-integer  
'data types. The problem with the  
populateComboBoxWithAll()  
'method is that it is limited in size to about 50 two  
dimensional  
'entries. In addition, commas and semi-colons create  
problems  
'and must be removed from the string during build.

'Finally, when the user clicks double clicks a label in the  
table,  
'code is executed that builds the input string for stored  
procedure  
'2-0-1-1-flanCountflanFilteredMishaps  
'which is the recordsource for the 2-0-1-2-frm-ViewMishaps  
form.

'this input string is then passed to the "view" form via a  
global  
'variable and the viewMishaps form is opened.

References:  
' - 2-0-1-2-Frm-ViewMishaps  
' - clFormWindow  
' - ez\_SizingFunctions  
' - GlobalDeclarations

#####

\*\*\*\*\*  
' FUNCTIONS  
\*\*\*\*\*

=====

Function/Sub Name:  
' - cboAircraft\_Change()  
' ... thru ...  
' - cboYear\_Change()

Description: The next 9 subroutines are used to mark the  
'form as dirty (needing an update). Saves the state of the  
'data (and size of the form)

'Input: None

'Output: None

References: None

=====

Private Sub cboAircraft\_Change()  
bUpdateNeeded = True  
End Sub

Private Sub cboClass\_Change()  
bUpdateNeeded = True  
End Sub

Private Sub cboFactors1\_Change()  
bUpdateNeeded = True  
End Sub

Private Sub cboFactors2\_Change()  
bUpdateNeeded = True  
End Sub

Private Sub cboFactors3\_Change()  
bUpdateNeeded = True  
End Sub

Private Sub cboLocation\_Change()  
bUpdateNeeded = True  
End Sub

Private Sub cboOrganization\_Change()  
bUpdateNeeded = True  
End Sub

```
Private Sub cboType_Change()
    bUpdateNeeded = True
End Sub
```

```
Private Sub cboYear_Change()
    bUpdateNeeded = True
End Sub
```

```
=====
Function/Sub Name: cmdClose_Click()
'
```

```
'Description: Closes the form.
'
```

```
'Input: None
'
```

```
'Output: None
'
```

```
'References: None
'
```

```
=====
Private Sub cmdClose_Click()
    DoCmd.Close acForm, "2-0-2-1-fm-Summary"
End Sub
```

```
=====
Function/Sub Name: cmdUpdate_Click()
'
```

```
'Description: Updates all data on the form by calling
goGetUpdate().
'
```

```
'Input: None
'
```

```
'Output: None
'
```

```
'References:
'    - goGetUpdate()
'
```

```
=====
Private Sub cmdUpdate_Click()
    goGetUpdate
End Sub
```

```
=====
Function/Sub Name: Form_Activate()
'
```

```
'Description: Update the menu bar.
'
```

```
'Input: None
'
```

```
'Output: None
'
```

```
'References: None
'
```

```
=====
Private Sub Form_Activate()
    Application.CommandBars("mnuOther").Visible = True
End Sub
```

```
=====
Function/Sub Name: Form_Close()
'
```

```
'Description: Closes the form.
'
```

```
'Input: None
'
```

```
'Output: None
'
```

```
'References: None
'
```

```
=====
Private Sub Form_Close()
    Application.CommandBars("mnuOther").Visible = False
End Sub
```

```
=====
Function/Sub Name: Form_Deactivate()
'
```

```
'Description: Updates the menu bar.
'
```

```
'Input: None
'
```

```
'Output: None
'
```

```
'References: None
'
```

```
=====
Private Sub Form_Deactivate()
    Application.CommandBars("mnuOther").Visible = False
End Sub
```

```
=====
Function/Sub Name: Form_Load()
'
```

```
'Description: Dynamically resizes the form to the users
screen
resolution and then centers it.
'
```

```
'Input: None
'
```

```
'Output: None
'
```

```
'References:
'    - ezSizeForm
'
```

```
=====
Private Sub Form_Load()
    'Dynamically resize the form based on screen resolution.
    ezSizeForm Me, -1
    MoveToCenter "2-0-2-1-fm-Summary"
End Sub
```

```
=====
Function/Sub Name: Form_Open()
'
```

```
'Description: Populates combo boxes. In order to allow the
combo
boxes to offer <All> as a choice, 2 methods are
needed -- one for integers and another for strings.
```

```
'The populateComboBoxWillAll() subroutine is used for
integers (like the Mishap
Year), while stored procedures are used for strings.
```

```
'Important to note that the populateComboBoxWillAll() will
not work for
creating strings of more than about 50 entries because the
combo box
```

```

'rejects them as too, long. Stored procedures, however, do
not suffer
'from this limitation.
'
'Input: None
'
'Output: None
'
'References:
'      - populateComboBoxWithAll()
'
'=====
Private Sub Form_Open(Cancel As Integer)

    Application.CommandBars("mnuOther").Visible = True

    'Aircraft
    With Me.cboAircraft
        .RowSource = "9-0-0-2-flanLookupAircraftAll"
        .Value = "<All>"
    End With

    'Organization
    With Me.cboOrganization
        .RowSource = "9-0-0-2-flanLookupOrganizationAll"
        .Value = "<All>"
    End With

    'Location
    With Me.cboLocation
        .RowSource = "9-0-0-2-flanLookupLocationAll"
        .Value = "<All>"
    End With

    'Class category
    With Me.cboClass
        .RowSource = "9-0-0-2-flanLookupClassAll"
        .Value = "<All>"
    End With

    'Type category
    With Me.cboType
        .RowSource = "9-0-0-2-flanLookupTypeAll"
        .Value = "<All>"
    End With

    'Year (can't use UNION stored procedure to append <All>
because it is of type Integer)
    populateComboBoxWithAll "9-0-0-2-
flanModifiedLookupYear", 1
    With Me.cboYear
        .RowSourceType = "Value List"
        .RowSource = tempvaluelist
        .Value = "<All>"
    End With

    '3rd Level factors
    With Me.cboFactors3
        .RowSource = "9-0-0-2-flanLookupFactorsAll3Level"
        .Value = "<All>"
    End With

    '2nd Level factors
    With Me.cboFactors2
        .RowSource = "9-0-0-2-flanLookupFactorsAll2Level"
        .Value = "<All>"
    End With

```

```

'1st Level factors
With Me.cboFactors1
    .RowSource = "9-0-0-2-flanLookupFactorsAll1Level"
    .Value = "<All>"
End With

'Reset the update variable (because) the form is not dirty.
bUpdateNeeded = False
storeValues 'Store the initial values of the combo boxes.

End Sub

'=====
'Function/Sub Name: populateComboBoxWithAll()
'
'Description: Makes a connection to the stored procedure
passed-in
'and builds an string that can be used by a combo box to
display
'and "iNumberColToGet" column drop down list. It has to
check
'every record for commas and semi-colons in the data
because these
'two characters are interpreted by the combo box as
delimiters,
'so they must be replaced with some other character (a "-" is
what
I'm using here).
'
'Input:
'      sNameOfSP      - Name of the Stored Procedure to
get
'                      the records from.
'
'      iNumberColToGet - Number of columns of data to
read
'                      from the Stored Procedure.
'
'Output: None
'
'References:
'      - clFormWindow
'
'=====
Private Sub populateComboBoxWithAll(sNameOfSP As
String, _
    iNumberColToGet As Integer)

    'STEP 1 - Make a connection and get a recordset matching
the
'passed in parameters
    Dim cnn As Connection
    Dim oCmd As ADODB.Command
    Dim rst As ADODB.Recordset
    Set cnn = CurrentProject.Connection
    cnn.CursorLocation = adUseClient
    Set rst = New ADODB.Recordset
    Set oCmd = New ADODB.Command
    oCmd.ActiveConnection = cnn
    oCmd.CommandText = """" & sNameOfSP & """"
    oCmd.CommandType = adCmdStoredProc
    Set rst = oCmd.Execute

    'Make sure tempvalue list is empty before adding to it.
    tempvaluelist = ""

```

```

'STEP 2 - Build a string of all the values starting with
choice <All>.
Dim i As Integer
i = 0
For i = 0 To (iNumberColToGet - 1)
    tempvaelist = "<All>," & tempvaelist 'Add <All>
Next

'Now add the real values
rst.MoveFirst
Do Until rst.EOF
    Dim k As Integer
    k = 0
    While k < iNumberColToGet

        'STEP 3 - Replace commas and semicolons with
dashes becuase
        'the mess up the list
        Dim astrText As String
        Dim iCount As Integer
        'Check for null fields and only operate on those that
are not null
        If IsNull(rst.Fields(k)) Then
            tempvaelist = tempvaelist & rst.Fields(k) & ","
        Else
            astrText = Trim(rst.Fields(k))

            ' Loop through array, replacing commas and
semicolons
            For iCount = 1 To Len(astrText)

                If Mid(astrText, iCount, 1) = "," Or
Mid(astrText, iCount, 1) = ";" Then
                    ' If array element satisfies wildcard search,
                    ' replace it.
                    Mid(astrText, iCount, 1) = "-"
                End If
            Next
            ' Join string.
            tempvaelist = tempvaelist & astrText & ","
        End If
        k = k + 1
    Wend
    rst.MoveNext
Loop

rst.Close
Set rst = Nothing
Set oCmd = Nothing
cnn.Close

End Sub

'=====
'Function/Sub Name: goGetUpdate()
'
'Description: Builds the input string to pass based on the
users
'combo box selection and uses this information to requery
'the underlying recordsource for this form. This updates the
'table to show the counts corresponding to the user's combo
box
'criteria.
'
'Input: None
'
'Output: None

```

```

'
'References:
'
' - gStrInputString
'
'=====
Private Sub goGetUpdate()
On Error GoTo Err_goGetUpdate

'Reset the global variable
GlobalDeclarations.gStrInputString = ""

'Build the input string to pass to the stored procedure
'to get the correct records.
'Order of these if statements must match the SP.
'If <All> was selected, then pass " so that the SP knows the
value is NULL.
If Me.cboAircraft.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString = "@AC_Type
varchar(10)=" & Me.cboAircraft.Value & ""
Else
    GlobalDeclarations.gStrInputString = "@AC_Type
varchar(10)=""
End If

If Me.cboType.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Mishap_Type
varchar(3)=" & Me.cboType.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Mishap_Type
varchar(3)=""
End If

If Me.cboClass.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Mishap_Class
varchar(1)=" & Me.cboClass.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Mishap_Class
varchar(1)=""
End If

If Me.cboLocation.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Location
varchar(25)=" & Me.cboLocation.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Location
varchar(25)=""
End If

If Me.cboOrganization.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Service
varchar(10)=" & Me.cboOrganization.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Service
varchar(10)=""
End If

If Me.cboYear.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Year int=" &
Me.cboYear.Value & ""

```



```

Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Year int="
End If

If Me.cboFactors1.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @1stLevel
varchar(5)=" & Me.cboFactors1.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @1stLevel
varchar(5)="
End If

If Me.cboFactors2.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @2ndLevel
varchar(5)=" & Me.cboFactors2.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @2ndLevel
varchar(5)="
End If

If Me.cboFactors3.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @3rdLevel
varchar(5)=" & Me.cboFactors3.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @3rdLevel
varchar(5)="
End If

Me.InputParameters = GlobalDeclarations.gStrInputString
Me.Requery 'Update the form.
bUpdateNeeded = False 'Reset the forms dirty variable.
storeValues

If Me.txtMishapTotal = 0 Then
    MsgBox "There are no records that match your search
criteria.", vbOKOnly + vbInformation, "Criteria Too
Restrictive"
End If

Exit_goGetUpdate:
Exit Sub

Err_goGetUpdate:
MsgBox ERR.Description
Resume Exit_goGetUpdate

End Sub

```

```

'=====
Function/Sub Name: goGetRecords()
'
'Description: Builds the input string to pass to the stored
procedure
'to get the correct records. Order of these if statements must
match the SP.
'If <All> was selected, then pass " so that the SP knows the
value is NULL.

```

```

'Once the input string is built, the 2-0-1-2-frm-ViewMishaps
form
'is opened.
'
'Input: None
'
'Output: None
'
'References:
'      - 2-0-1-2-frm-ViewMishaps
'      - gStrInputString
'=====
Private Sub goGetRecords()
On Error GoTo Err_goGetRecords

'Reset the global variable
GlobalDeclarations.gStrInputString = ""

'Build the input string to pass to the stored procedure to get
the correct records.
'Order of these if statements must match the SP.
'If <All> was selected, then pass " so that the SP knows the
value is NULL.
If Me.cboAircraft.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString = "@AC
varchar(10)=" & Me.cboAircraft.Value & ""
Else
    GlobalDeclarations.gStrInputString = "@AC
varchar(10)="
End If

If Me.cboType.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Type
varchar(3)=" & Me.cboType.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Type
varchar(3)="
End If

If Me.cboClass.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Class
varchar(1)=" & Me.cboClass.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Class
varchar(1)="
End If

If Me.cboLocation.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Loc
varchar(25)=" & Me.cboLocation.Value & ""
Else
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Loc
varchar(25)="
End If

If Me.cboOrganization.Value <> "<All>" Then
    GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Svc
varchar(10)=" & Me.cboOrganization.Value & ""
Else

```

```

GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Svc
varchar(10)="
End If

If Me.cboYear.Value <> "<All>" Then
GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Yr int=" &
Me.cboYear.Value & ""
Else
GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @Yr int="
End If

If Me.cboFactors1.Value <> "<All>" Then
GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @1stLevel
varchar(5)=" & Me.cboFactors1.Value & ""
Else
GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @1stLevel
varchar(5)="
End If

If Me.cboFactors2.Value <> "<All>" Then
GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @2ndLevel
varchar(5)=" & Me.cboFactors2.Value & ""
Else
GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @2ndLevel
varchar(5)="
End If

If Me.cboFactors3.Value <> "<All>" Then
GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @3rdLevel
varchar(5)=" & Me.cboFactors3.Value & ""
Else
GlobalDeclarations.gStrInputString =
GlobalDeclarations.gStrInputString & ", @3rdLevel
varchar(5)="
End If

'Set flag to tell the ViewMishaps form to use the correct
SP for viewing factor category recordsets.
GlobalDeclarations.bUseHFACSSummaryQuery = True
DoCmd.OpenForm "2-0-1-2-fm-ViewMishaps"

Exit_goGetRecords:
Exit Sub

Err_goGetRecords:
MsgBox ERR.Description
Resume Exit_goGetRecords

End Sub

'=====
'Function/Sub Name: goGetRecords()
'
'Description: Store the values of the filter boxes on form open
and
'after every update so that you have something to compare
current values to.
'This way, you can trap when users make changes.

```

```

'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub storeValues()

sStoredAircraft = Me.cboAircraft.Value
sStoredType = Me.cboType.Value
sStoredClass = Me.cboClass.Value
sStoredLocation = Me.cboLocation.Value
sStoredOrganization = Me.cboOrganization.Value
vStoredYear = Me.cboYear.Value
sStored1stLevel = Me.cboFactors1.Value
sStored2ndLevel = Me.cboFactors2.Value
sStored3rdLevel = Me.cboFactors3.Value

End Sub

'=====
'Function/Sub Name: checkIfFormIsDirty()
'
'Description: If the user changed values in the combo boxes
but has not
'updated the form, tell him about it and give the option to
refresh
'before viewing records. If you don't do this, then the user
might
'change the combo box criteria and then forget to hit the
update
'button before double-clicking one of the boxes. This could
create
'confusing results.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub checkIfFormIsDirty()

If bUpdateNeeded = True Then

If sStoredAircraft <> Me.cboAircraft.Value Or _
sStoredType <> Me.cboType.Value Or _
sStoredClass <> Me.cboClass.Value Or _
sStoredLocation <> Me.cboLocation.Value Or _
sStoredOrganization <> Me.cboOrganization.Value
Or _
vStoredYear <> Me.cboYear.Value Or _
sStored1stLevel <> Me.cboFactors1.Value Or _
sStored2ndLevel <> Me.cboFactors2.Value Or _
sStored3rdLevel <> Me.cboFactors3.Value Then

Dim response As Variant
response = MsgBox("You have changed selection
criteria but not clicked the UPDATE button to refresh the
data." & Chr(13) & Chr(13) & "Do you want to update the
data with the new criteria?", vbYesNo + vbQuestion +
vbDefaultButton1, "Form Needs Update")
If response = vbYes Then
goGetUpdate

```

```

        Else 'Set the comboboxes to the old values.
            Me.cboAircraft.Value = sStoredAircraft
            Me.cboType.Value = sStoredType
            Me.cboClass.Value = sStoredClass
            Me.cboLocation.Value = sStoredLocation
            Me.cboOrganization.Value =
sStoredOrganization
            Me.cboYear.Value = vStoredYear
            Me.cboFactors1.Value = sStored1stLevel
            Me.cboFactors2.Value = sStored2ndLevel
            Me.cboFactors3.Value = sStored3rdLevel
        End If
    End If
End Sub

```

Function/Sub Name:

```

' - lblADA_DblClick()
' ... thru ...
' - txtPRES_DblClick()
'

```

Description: Private subs-for detecting box double clicks follow.  
Three subroutines are needed for each box. One for the label and one form each text box (number and percentage).

Input: None

Output: None

References: None

```

Private Sub lblADA_DblClick(Cancel As Integer)
    If Me.txtADA.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "ADA"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblASS_DblClick(Cancel As Integer)
    If Me.txtASS.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "ASS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblATT_DblClick(Cancel As Integer)
    If Me.txtATT.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    End Sub

```

```

    Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors3.Value = "ATT"
goGetRecords
Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblCON_DblClick(Cancel As Integer)
    If Me.txtCON.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "CON"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblCRT_DblClick(Cancel As Integer)
    If Me.txtCRT.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "CRT"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblCRW_DblClick(Cancel As Integer)
    If Me.txtCRW.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "CRW"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub lblDES_DblClick(Cancel As Integer)
    If Me.txtDES.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "DES"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblDMG_DblClick(Cancel As Integer)
    If Me.txtDMG.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    Exit Sub
    End If
    checkIfFormIsDirty

```

```

    Me.cboFactors3.Value = "DMG"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblDOC_Db1Click(Cancel As Integer)
    If Me.txtDOC.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "DOC"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblDUC_Db1Click(Cancel As Integer)
    If Me.txtDUC.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "DUC"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblEHZ_Db1Click(Cancel As Integer)
    If Me.txtEHZ.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "EHZ"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblENV_Db1Click(Cancel As Integer)
    If Me.txtENV.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "ENV"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub lblEQP_Db1Click(Cancel As Integer)
    If Me.txtEQP.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "EQP"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel

```

```
End Sub
```

```

Private Sub lblERR_Db1Click(Cancel As Integer)
    If Me.txtERR.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "ERR"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub lblEXC_Db1Click(Cancel As Integer)
    If Me.txtEXC.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "EXC"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblFLG_Db1Click(Cancel As Integer)
    If Me.txtFLG.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "FLG"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblIDQ_Db1Click(Cancel As Integer)
    If Me.txtIDQ.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "IDQ"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub lblINA_Db1Click(Cancel As Integer)
    If Me.txtINA.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "INA"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```
Private Sub lblINF_Db1Click(Cancel As Integer)
```

```

    If Me.txtINF.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "INF"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblJDG_DblClick(Cancel As Integer)
    If Me.txtJDG.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "JDG"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblLGT_DblClick(Cancel As Integer)
    If Me.txtLGT.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "LGT"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblLIM_DblClick(Cancel As Integer)
    If Me.txtLIM.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "LIM"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblMA_DblClick(Cancel As Integer)
    If Me.txtMA.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors1.Value = "MA"
    goGetRecords
    Me.cboFactors1.Value = sStored1stLevel
End Sub

Private Sub lblMC_DblClick(Cancel As Integer)
    If Me.txtMC.Value = 0 Then

```

```

        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors1.Value = "MC"
    goGetRecords
    Me.cboFactors1.Value = sStored1stLevel
End Sub

Private Sub lblMED_DblClick(Cancel As Integer)
    If Me.txtMED.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "MED"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub lblMG_DblClick(Cancel As Integer)
    If Me.txtMG.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors1.Value = "MG"
    goGetRecords
    Me.cboFactors1.Value = sStored1stLevel
End Sub

Private Sub lblMIS_DblClick(Cancel As Integer)
    If Me.txtMIS.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "MIS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblMNT_DblClick(Cancel As Integer)
    If Me.txtMNT.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "MNT"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblRDY_DblClick(Cancel As Integer)
    If Me.txtRDY.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"

```

```

        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "RDY"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub lblROU_DblClick(Cancel As Integer)
    If Me.txtROU.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "ROU"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblSKL_DblClick(Cancel As Integer)
    If Me.txtSKL.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "SKL"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblSUP_DblClick(Cancel As Integer)
    If Me.txtSUP.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "SUP"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub lblTRG_DblClick(Cancel As Integer)
    If Me.txtTRG.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "TRG"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblPHY_DblClick(Cancel As Integer)
    If Me.txtPHY.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty

```

```

    Me.cboFactors3.Value = "PHY"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblCOM_DblClick(Cancel As Integer)
    If Me.txtCOM.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "COM"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblVIO_DblClick(Cancel As Integer)
    If Me.txtVIO.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "VIO"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub lblWC_DblClick(Cancel As Integer)
    If Me.txtWC.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors1.Value = "WC"
    goGetRecords
    Me.cboFactors1.Value = sStored1stLevel
End Sub

Private Sub lblWRK_DblClick(Cancel As Integer)
    If Me.txtWRK.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "WRK"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub lblWXE_DblClick(Cancel As Integer)
    If Me.txtWXE.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "WXE"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel

```

```

End Sub

Private Sub lblKNW_DblClick(Cancel As Integer)
    If Me.txtKNW.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "KNW"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblIFC_DblClick(Cancel As Integer)
    If Me.txtIFC.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "IFC"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblOBS_DblClick(Cancel As Integer)
    If Me.txtOBS.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "OBS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblUNA_DblClick(Cancel As Integer)
    If Me.txtUNA.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "UNA"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblOPS_DblClick(Cancel As Integer)
    If Me.txtOPS.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "OPS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblORG_DblClick(Cancel As Integer)

```

```

    If Me.txtORG.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "ORG"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub lblPRB_DblClick(Cancel As Integer)
    If Me.txtPRB.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "PRB"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblPRO_DblClick(Cancel As Integer)
    If Me.txtPRO.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "PRO"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub lblRES_DblClick(Cancel As Integer)
    If Me.txtRES.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "RES"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

'Number text boxes start here.

Private Sub txtADA_DblClick(Cancel As Integer)
    If Me.txtADA.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "ADA"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtASS_DblClick(Cancel As Integer)
    If Me.txtASS.Value = 0 Then

```

```

        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "ASS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtATT_Db1Click(Cancel As Integer)
    If Me.txtATT.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "ATT"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtCON_Db1Click(Cancel As Integer)
    If Me.txtCON.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "CON"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtCRT_Db1Click(Cancel As Integer)
    If Me.txtCRT.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "CRT"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtCRW_Db1Click(Cancel As Integer)
    If Me.txtCRW.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "CRW"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub txtDES_Db1Click(Cancel As Integer)
    If Me.txtDES.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    End If
    checkIfFormIsDirty

```

```

        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "DES"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtDMG_Db1Click(Cancel As Integer)
    If Me.txtDMG.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "DMG"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtDOC_Db1Click(Cancel As Integer)
    If Me.txtDOC.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "DOC"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtDUC_Db1Click(Cancel As Integer)
    If Me.txtDUC.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "DUC"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtEHZ_Db1Click(Cancel As Integer)
    If Me.txtEHZ.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "EHZ"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtENV_Db1Click(Cancel As Integer)
    If Me.txtENV.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty

```



```

        Me.cboFactors2.Value = "ENV"
        goGetRecords
        Me.cboFactors2.Value = sStored2ndLevel
    End Sub

    Private Sub txtEQP_DblClick(Cancel As Integer)
        If Me.txtEQP.Value = 0 Then
            MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
            Exit Sub
        End If
        checkIfFormIsDirty
        Me.cboFactors2.Value = "EQP"
        goGetRecords
        Me.cboFactors2.Value = sStored2ndLevel
    End Sub

    Private Sub txtERR_DblClick(Cancel As Integer)
        If Me.txtERR.Value = 0 Then
            MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
            Exit Sub
        End If
        checkIfFormIsDirty
        Me.cboFactors2.Value = "ERR"
        goGetRecords
        Me.cboFactors2.Value = sStored2ndLevel
    End Sub

    Private Sub txtEXC_DblClick(Cancel As Integer)
        If Me.txtEXC.Value = 0 Then
            MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
            Exit Sub
        End If
        checkIfFormIsDirty
        Me.cboFactors3.Value = "EXC"
        goGetRecords
        Me.cboFactors3.Value = sStored3rdLevel
    End Sub

    Private Sub txtFLG_DblClick(Cancel As Integer)
        If Me.txtFLG.Value = 0 Then
            MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
            Exit Sub
        End If
        checkIfFormIsDirty
        Me.cboFactors3.Value = "FLG"
        goGetRecords
        Me.cboFactors3.Value = sStored3rdLevel
    End Sub

    Private Sub txtIDQ_DblClick(Cancel As Integer)
        If Me.txtIDQ.Value = 0 Then
            MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
            Exit Sub
        End If
        checkIfFormIsDirty
        Me.cboFactors3.Value = "IDQ"
        goGetRecords
        Me.cboFactors3.Value = sStored3rdLevel
    End Sub

```

```

    End Sub

    Private Sub txtINA_DblClick(Cancel As Integer)
        If Me.txtINA.Value = 0 Then
            MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
            Exit Sub
        End If
        checkIfFormIsDirty
        Me.cboFactors3.Value = "INA"
        goGetRecords
        Me.cboFactors3.Value = sStored3rdLevel
    End Sub

    Private Sub txtINF_DblClick(Cancel As Integer)
        If Me.txtINF.Value = 0 Then
            MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
            Exit Sub
        End If
        checkIfFormIsDirty
        Me.cboFactors3.Value = "INF"
        goGetRecords
        Me.cboFactors3.Value = sStored3rdLevel
    End Sub

    Private Sub txtJDG_DblClick(Cancel As Integer)
        If Me.txtJDG.Value = 0 Then
            MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
            Exit Sub
        End If
        checkIfFormIsDirty
        Me.cboFactors3.Value = "JDG"
        goGetRecords
        Me.cboFactors3.Value = sStored3rdLevel
    End Sub

    Private Sub txtLGT_DblClick(Cancel As Integer)
        If Me.txtLGT.Value = 0 Then
            MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
            Exit Sub
        End If
        checkIfFormIsDirty
        Me.cboFactors3.Value = "LGT"
        goGetRecords
        Me.cboFactors3.Value = sStored3rdLevel
    End Sub

    Private Sub txtLIM_DblClick(Cancel As Integer)
        If Me.txtLIM.Value = 0 Then
            MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
            Exit Sub
        End If
        checkIfFormIsDirty
        Me.cboFactors3.Value = "LIM"
        goGetRecords
        Me.cboFactors3.Value = sStored3rdLevel
    End Sub

    Private Sub txtMA_DblClick(Cancel As Integer)

```

```

If Me.txtMA.Value = 0 Then
    MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors1.Value = "MA"
goGetRecords
Me.cboFactors1.Value = sStored1stLevel
End Sub

```

```

Private Sub txtMC_DblClick(Cancel As Integer)
    If Me.txtMC.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors1.Value = "MC"
    goGetRecords
    Me.cboFactors1.Value = sStored1stLevel
End Sub

```

```

Private Sub txtMED_DblClick(Cancel As Integer)
    If Me.txtMED.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "MED"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub txtMG_DblClick(Cancel As Integer)
    If Me.txtMG.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors1.Value = "MG"
    goGetRecords
    Me.cboFactors1.Value = sStored1stLevel
End Sub

```

```

Private Sub txtMIS_DblClick(Cancel As Integer)
    If Me.txtMIS.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "MIS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtMNT_DblClick(Cancel As Integer)
    If Me.txtMNT.Value = 0 Then

```

```

        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "MNT"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPROU_DblClick(Cancel As Integer)
    If Me.txtROU.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "ROU"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPSKL_DblClick(Cancel As Integer)
    If Me.txtSKL.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "SKL"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtRDY_DblClick(Cancel As Integer)
    If Me.txtRDY.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "RDY"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub txtROU_DblClick(Cancel As Integer)
    If Me.txtROU.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "ROU"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtSKL_DblClick(Cancel As Integer)
    If Me.txtSKL.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"

```

```

Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors3.Value = "SKL"
goGetRecords
Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtSUP_DblClick(Cancel As Integer)
If Me.txtSUP.Value = 0 Then
MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors2.Value = "SUP"
goGetRecords
Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub txtTRG_DblClick(Cancel As Integer)
If Me.txtTRG.Value = 0 Then
MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors3.Value = "TRG"
goGetRecords
Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPHY_DblClick(Cancel As Integer)
If Me.txtPHY.Value = 0 Then
MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors3.Value = "PHY"
goGetRecords
Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtCOM_DblClick(Cancel As Integer)
If Me.txtCOM.Value = 0 Then
MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors3.Value = "COM"
goGetRecords
Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtVIO_DblClick(Cancel As Integer)
If Me.txtVIO.Value = 0 Then
MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
Exit Sub
End If
checkIfFormIsDirty

```

```

Me.cboFactors2.Value = "VIO"
goGetRecords
Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub txtWC_DblClick(Cancel As Integer)
If Me.txtWC.Value = 0 Then
MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors1.Value = "WC"
goGetRecords
Me.cboFactors1.Value = sStored1stLevel
End Sub

Private Sub txtWRK_DblClick(Cancel As Integer)
If Me.txtWRK.Value = 0 Then
MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors2.Value = "WRK"
goGetRecords
Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub txtWXE_DblClick(Cancel As Integer)
If Me.txtWXE.Value = 0 Then
MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors3.Value = "WXE"
goGetRecords
Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtKNW_DblClick(Cancel As Integer)
If Me.txtKNW.Value = 0 Then
MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors3.Value = "KNW"
goGetRecords
Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtIFC_DblClick(Cancel As Integer)
If Me.txtIFC.Value = 0 Then
MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors3.Value = "IFC"
goGetRecords
Me.cboFactors3.Value = sStored3rdLevel

```

```

End Sub

Private Sub txtOBS_DblClick(Cancel As Integer)
    If Me.txtOBS.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "OBS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtUNA_DblClick(Cancel As Integer)
    If Me.txtUNA.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "UNA"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtOPS_DblClick(Cancel As Integer)
    If Me.txtOPS.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "OPS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtORG_DblClick(Cancel As Integer)
    If Me.txtORG.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "ORG"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub txtPRB_DblClick(Cancel As Integer)
    If Me.txtPRB.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "PRB"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPRO_DblClick(Cancel As Integer)

```

```

    If Me.txtPRO.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "PRO"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtRES_DblClick(Cancel As Integer)
    If Me.txtRES.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "RES"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

'Percentage textboxes start here.

```

Private Sub txtPADA_DblClick(Cancel As Integer)
    If Me.txtADA.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "ADA"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPASS_DblClick(Cancel As Integer)
    If Me.txtASS.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "ASS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPATT_DblClick(Cancel As Integer)
    If Me.txtATT.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "ATT"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPCON_DblClick(Cancel As Integer)
    If Me.txtCON.Value = 0 Then

```

```

        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "CON"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPCRT_DblClick(Cancel As Integer)
    If Me.txtCRT.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "CRT"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPCRW_DblClick(Cancel As Integer)
    If Me.txtCRW.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "CRW"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub txtPDES_DblClick(Cancel As Integer)
    If Me.txtDES.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "DES"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPDMG_DblClick(Cancel As Integer)
    If Me.txtDMG.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "DMG"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPDOG_DblClick(Cancel As Integer)
    If Me.txtDOC.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    End If
    checkIfFormIsDirty

```

```

        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "DOC"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPDUC_DblClick(Cancel As Integer)
    If Me.txtDUC.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "DUC"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPEHZ_DblClick(Cancel As Integer)
    If Me.txtEHZ.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "EHZ"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPENV_DblClick(Cancel As Integer)
    If Me.txtENV.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "ENV"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub txtPEQP_DblClick(Cancel As Integer)
    If Me.txtEQP.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "EQP"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub txtPERR_DblClick(Cancel As Integer)
    If Me.txtERR.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty

```

```

        Me.cboFactors2.Value = "ERR"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub txtPEXC_DblClick(Cancel As Integer)
    If Me.txtEXC.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "EXC"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPFLG_DblClick(Cancel As Integer)
    If Me.txtFLG.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "FLG"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPIDQ_DblClick(Cancel As Integer)
    If Me.txtIDQ.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "IDQ"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPINA_DblClick(Cancel As Integer)
    If Me.txtINA.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "INA"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPINF_DblClick(Cancel As Integer)
    If Me.txtINF.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "INF"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

End Sub

Private Sub txtPIDG_DblClick(Cancel As Integer)
    If Me.txtJDG.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "JDG"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPLGT_DblClick(Cancel As Integer)
    If Me.txtLGT.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "LGT"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPLIM_DblClick(Cancel As Integer)
    If Me.txtLIM.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "LIM"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPMA_DblClick(Cancel As Integer)
    If Me.txtMA.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors1.Value = "MA"
    goGetRecords
    Me.cboFactors1.Value = sStored1stLevel
End Sub

Private Sub txtPMC_DblClick(Cancel As Integer)
    If Me.txtMC.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors1.Value = "MC"
    goGetRecords
    Me.cboFactors1.Value = sStored1stLevel
End Sub

Private Sub txtPMED_DblClick(Cancel As Integer)

```

```

If Me.txtMED.Value = 0 Then
    MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
    Exit Sub
End If
checkIfFormIsDirty
Me.cboFactors2.Value = "MED"
goGetRecords
Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub txtPMG_DblClick(Cancel As Integer)
    If Me.txtMG.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors1.Value = "MG"
    goGetRecords
    Me.cboFactors1.Value = sStored1stLevel
End Sub

```

```

Private Sub txtPMIS_DblClick(Cancel As Integer)
    If Me.txtMIS.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "MIS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPMNT_DblClick(Cancel As Integer)
    If Me.txtMNT.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "MNT"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPRDY_DblClick(Cancel As Integer)
    If Me.txtRDY.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "RDY"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub txtPSUP_DblClick(Cancel As Integer)
    If Me.txtSUP.Value = 0 Then

```

```

        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "SUP"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub txtPTRG_DblClick(Cancel As Integer)
    If Me.txtTRG.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "TRG"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPPHY_DblClick(Cancel As Integer)
    If Me.txtPHY.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "PHY"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPCOM_DblClick(Cancel As Integer)
    If Me.txtCOM.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "COM"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

```

```

Private Sub txtPVIO_DblClick(Cancel As Integer)
    If Me.txtVIO.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "VIO"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

```

```

Private Sub txtPWC_DblClick(Cancel As Integer)
    If Me.txtWC.Value = 0 Then
        MsgBox "There are no records in that category to
view.", vbOKOnly + vbExclamation, "Criteria Too
Restrictive"

```

```

        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors1.Value = "WC"
    goGetRecords
    Me.cboFactors1.Value = sStored1stLevel
End Sub

Private Sub txtPWRK_DblClick(Cancel As Integer)
    If Me.txtWRK.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "WRK"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub txtPWXE_DblClick(Cancel As Integer)
    If Me.txtWXE.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "WXE"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPKNW_DblClick(Cancel As Integer)
    If Me.txtKNW.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "KNW"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPIFC_DblClick(Cancel As Integer)
    If Me.txtIFC.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "IFC"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPOBS_DblClick(Cancel As Integer)
    If Me.txtOBS.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty

```

```

    Me.cboFactors3.Value = "OBS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPUNA_DblClick(Cancel As Integer)
    If Me.txtUNA.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "UNA"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPOPS_DblClick(Cancel As Integer)
    If Me.txtOPS.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "OPS"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPORG_DblClick(Cancel As Integer)
    If Me.txtORG.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors2.Value = "ORG"
    goGetRecords
    Me.cboFactors2.Value = sStored2ndLevel
End Sub

Private Sub txtPPRB_DblClick(Cancel As Integer)
    If Me.txtPRB.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "PRB"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

Private Sub txtPPRO_DblClick(Cancel As Integer)
    If Me.txtPRO.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "PRO"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel

```



```

End Sub

Private Sub txtPRES_DblClick(Cancel As Integer)
    If Me.txtRES.Value = 0 Then
        MsgBox "There are no records in that category to view.", vbOKOnly + vbExclamation, "Criteria Too Restrictive"
        Exit Sub
    End If
    checkIfFormIsDirty
    Me.cboFactors3.Value = "RES"
    goGetRecords
    Me.cboFactors3.Value = sStored3rdLevel
End Sub

'=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the ezSizeMode
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of this

```

```

'function so that minor adjustments can be made on a form by
'form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'         - clFormWindow
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        .Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) * 0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

End Sub

```

## **FORMCLASS-4-0-1-0-fm-ExpertGraph**

Option Compare Database

Option Explicit

'#####

### FORM DESCRIPTION

'#####

'Class Name: 4-0-1-0-fm-Expert Graph

'

'Author: Pat Flanders & Scott Tufts

'

'This class is used to select the X and Y axis criteria and pass

'the users selections to global variables that the

'4-0-1-2-fm-TheActualGraph can use to display the graph.

'

'References:

' - 4-0-1-2-fm-TheActualGraph

' - clFormWindow

' - ez\_SizingFunctions

' - GlobalDeclarations

'

'#####

\*\*\*\*\*

### FUNCTIONS

\*\*\*\*\*

'=====

'Function/Sub Name:

' - cmdClose\_MouseMove()

' ... thru ...

' - cmdGraph\_MouseMove()

'

'Description: Changes the color of the command button text

'in response to a mouse move event.

'

'Input: None

'

'Output: None

'

'References: None

'

'=====

Private Sub cmdClose\_MouseMove(Button As Integer, Shift  
As Integer, X As Single, Y As Single)

' Make the button text blue when it gets the focus

Me.cmdGraph.ForeColor = QBColor(0)

Me.cmdClose.ForeColor = QBColor(9)

End Sub

Private Sub cmdGraph\_MouseMove(Button As Integer, Shift  
As Integer, X As Single, Y As Single)

' Make the button text blue when it gets the focus

Me.cmdGraph.ForeColor = QBColor(9)

Me.cmdClose.ForeColor = QBColor(0)

End Sub

'=====

'Function/Sub Name: cmdGraph\_Click()

'

'Description: Passes the appropriate field names

corresponding to

'user choices for X and Y axis graph criteria to global

variables

'for the 4-0-1-2-fm-TheActualGraph form to actually create

the

'graph.

'

'Input: None

'

'Output: None

'

'References:

' - 4-0-1-2-fm-TheActualGraph

' - GlobalDeclarations.gStrXFieldToGraph

' - GlobalDeclarations.gStrYFieldToGraph

'

'=====

Private Sub cmdGraph\_Click()

If Me.fraX.Value = Me.fraY.Value Then

MsgBox "Your selections for the X and Y axis must be  
different.", vbOKOnly + vbExclamation, "Choose Different  
Values"

Exit Sub

End If

Select Case Me.fraX.Value

Case 1 'Aircraft

If Me.chkUseCodesX.Value = True Then

GlobalDeclarations.gStrXFieldToGraph =

"Aircraft\_FK"

Else

GlobalDeclarations.gStrXFieldToGraph =

"Aircraft\_FK"

End If

Case 2 'Organization

If Me.chkUseCodesX.Value = True Then

GlobalDeclarations.gStrXFieldToGraph =

"OrgID\_FK"

Else

GlobalDeclarations.gStrXFieldToGraph =

"OrgName"

End If

Case 3 'Location

If Me.chkUseCodesX.Value = True Then

GlobalDeclarations.gStrXFieldToGraph =

"LocationID\_FK"

Else

GlobalDeclarations.gStrXFieldToGraph =

"MishapLocation"

End If

Case 4 'Class

If Me.chkUseCodesX.Value = True Then

GlobalDeclarations.gStrXFieldToGraph =

"Class\_FK"

Else

GlobalDeclarations.gStrXFieldToGraph =

"MishapClassDefinition"

End If

Case 5 'Type

If Me.chkUseCodesX.Value = True Then

GlobalDeclarations.gStrXFieldToGraph =

"Type\_FK"

```

Else
    GlobalDeclarations.gStrXFieldToGraph =
"MishapTypeDefinition"
End If
Case 6 'Year
    If Me.chkUseCodesX.Value = True Then
        GlobalDeclarations.gStrXFieldToGraph = "Year"
    Else
        GlobalDeclarations.gStrXFieldToGraph = "Year"
    End If
End Select

Select Case Me.fraY.Value
Case 1 'Aircraft
    If Me.chkUseCodesY.Value = True Then
        GlobalDeclarations.gStrYFieldToGraph =
"Aircraft_FK"
    Else
        GlobalDeclarations.gStrYFieldToGraph =
"Aircraft_FK"
    End If
Case 2 'Organization
    If Me.chkUseCodesY.Value = True Then
        GlobalDeclarations.gStrYFieldToGraph =
"OrgID_FK"
    Else
        GlobalDeclarations.gStrYFieldToGraph =
"OrgName"
    End If
Case 3 'Location
    If Me.chkUseCodesY.Value = True Then
        GlobalDeclarations.gStrYFieldToGraph =
"LocationID_FK"
    Else
        GlobalDeclarations.gStrYFieldToGraph =
"MishapLocation"
    End If
Case 4 'Class
    If Me.chkUseCodesY.Value = True Then
        GlobalDeclarations.gStrYFieldToGraph =
"Class_FK"
    Else
        GlobalDeclarations.gStrYFieldToGraph =
"MishapClassDefinition"
    End If
Case 5 'Type
    If Me.chkUseCodesY.Value = True Then
        GlobalDeclarations.gStrYFieldToGraph =
"Type_FK"
    Else
        GlobalDeclarations.gStrYFieldToGraph =
"MishapTypeDefinition"
    End If
Case 6 'Year
    If Me.chkUseCodesY.Value = True Then
        GlobalDeclarations.gStrYFieldToGraph = "Year"
    Else
        GlobalDeclarations.gStrYFieldToGraph = "Year"
    End If
End Select

DoCmd.OpenForm "7-0-0-1-PopUpFrm-
waitProgressbar", acNormal, "", "", acReadOnly, acNormal
DoCmd.RepaintObject acForm, "7-0-0-1-PopUpFrm-
waitProgressbar"
DoCmd.OpenForm "4-0-1-2-fm-TheActualGraph"
DoCmd.Close acForm, "7-0-0-1-PopUpFrm-
waitProgressbar"

```

End Sub

```

=====
'Function/Sub Name: Form_Activate()
'
'Description: Update the menu bar.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Form_Activate()
    Application.CommandBars("mnuOther").Visible = True
End Sub

=====
'Function/Sub Name: Form_Deactivate()
'
'Description: Updates the menu bar.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Form_Deactivate()
    Application.CommandBars("mnuOther").Visible = False
End Sub

=====
'Function/Sub Name: Form_Close()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Form_Close()

    Application.CommandBars("mnuOther").Visible = False
    Application.CommandBars("mnuProgramMain").Visible =
True
    Forms![MainMenu].Visible = True

End Sub

=====
'Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None

```

```

'
'Output: None
'
'References:
'    - ezSizeForm
'
'=====
Private Sub Form_Load()
    ezSizeForm Me, -1
    MoveToCenter "4-0-1-0-fm-ExpertGraph"
End Sub

'=====
'Function/Sub Name: Form_Open()
'
'Description: Updates the menu bar and sets the focus to the
'close button.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Open(Cancel As Integer)

    Forms![MainMenu].Visible = False
    Application.CommandBars("mnuOther").Visible = True

    ' Make the button text blue when it gets the focus
    Me.cmdGraph.ForeColor = QBColor(0)
    Me.cmdClose.ForeColor = QBColor(0)

    Me.cmdClose.SetFocus

End Sub

'=====
'Function/Sub Name: cmdClose_Click
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None

```

```

'
'=====
Private Sub cmdClose_Click()
    On Error GoTo Err_cmdClose_Click

    DoCmd.Close acForm, "4-0-1-0-fm-ExpertGraph"

Exit_cmdClose_Click:
    Exit Sub

Err_cmdClose_Click:
    MsgBox ERR.Description
    Resume Exit_cmdClose_Click

End Sub

'=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'    - clFormWindow
'
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        '.Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

End Sub

```

## **FORMCLASS-4-0-1-2-fm-TheActualGraph**

```
Option Compare Database
Option Explicit

'Reusable variable for opening a connection
Dim cnn As Connection

'Reusable variable for opening a connection used in
conjunction
'with cnn.
Dim oCmd As ADODB.Command

'Reusable variable for recordset operations
Dim rst As ADODB.Recordset

'Arrays for storing user selections from the "Select X Values"
'and "Select Y Values" list boxes.
Dim aryItemsSelectedX() As Integer
Dim aryItemsSelectedY() As Integer

'#####
'          FORM DESCRIPTION
'#####
'Class Name: 4-0-1-2-fm-TheActualGraph
'
'Author: Pat Flanders & Scott Tufts
'
'Description: Uses the MSChart20 Active-X control to create
a
'graph based upon globalvariables passed from the
'4-0-1-0-fm-ExpertGraph form.
'
'The MSChart20 control creates a graph based upon values in
its
'DataGrid. The datagrid is not visible and must be populated
'completely via code. Various methods in this class are used
'to populate the datagrid and then show portions of it based
'on input from the user.
'
'The datagrid data is obtained from the RAC (Replacement
For
'Access Crosstab) stored procedures to create the crosstab
results
'based on the values of
GlobalDeclarations.gStrXFieldToGraph and
'GlobalDeclarations.gStrYFieldToGraph
'
'References:
'      - MSChart2.0 Active X control.
'      - clFormWindow
'      - ez_SizingFunctions
'      - GlobalDeclarations
'#####

*****
'          FUNCTIONS
*****

'=====
Function/Sub Name: chkStack_AfterUpdate()
'
'Description: Sets the Stacking option of the MSChart
control
'in response to a checkbox update.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub chkStack_AfterUpdate()

'Turn stacking on or off
If Me.chkStack.Value = True Then
    Me.chtTheGraph.Stacking = True
Else
    Me.chtTheGraph.Stacking = False
End If

End Sub

'=====
Function/Sub Name: chkTranspose_AfterUpdate()
'
'Description: Sets the DataSeriesInRow option of the
MSChart control
'in response to a checkbox update.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub chkTranspose_AfterUpdate()

'Turn transpose of X and Y axis on or off
If Me.chkTranspose.Value = True Then
    Me.chtTheGraph.Plot.DataSeriesInRow = True
Else
    Me.chtTheGraph.Plot.DataSeriesInRow = False
End If

End Sub

'=====
Function/Sub Name: chtTheGraph_LostFocus()
'
'Description: Updates the "Tips" label with information for
the
'user.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
```

```

Private Sub chtTheGraph_LostFocus()
    Me.lblTips.Caption = "Select a point to see its value."
End Sub

'=====
'Function/Sub Name: cmdClose_Click()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub cmdClose_Click()
On Error GoTo Err_cmdClose_Click

    DoCmd.Close

Exit_cmdClose_Click:
    Exit Sub

Err_cmdClose_Click:
    MsgBox ERR.Description
    Resume Exit_cmdClose_Click

End Sub

'=====
'Function/Sub Name: cmdUpdate_Click()
'
'Description: Rebuilds the MSChart20 control's Datagrid
based upon
'lstShowTheseX_AfterUpdate() and
'lstShowTheseY_AfterUpdate() information
'(which corresponds to the users selections in the X and Y
axis
'list box selection criteria).
'
'HINT: Uncomment the debug.print lines to troubleshoot this
code.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub cmdUpdate_Click()

    MsgBox strValueList

    'Go to the beginning of the recordset.
    rst.MoveFirst

    Me.chtTheGraph.DataGrid.RowCount =
Me.lstShowTheseX.ItemsSelected.Count
    'Debug.Print "RowCount = " &
Me.lstShowTheseX.ItemsSelected.Count
    Me.chtTheGraph.DataGrid.RowLabelCount =
Me.lstShowTheseX.ItemsSelected.Count + 1

```

```

    'Debug.Print "RowLabelCount = " &
Me.lstShowTheseX.ItemsSelected.Count + 1
    Me.chtTheGraph.DataGrid.ColumnCount =
Me.lstShowTheseY.ItemsSelected.Count
    'Debug.Print "ColumnCount = " &
Me.lstShowTheseY.ItemsSelected.Count
    Me.chtTheGraph.DataGrid.ColumnLabelCount =
Me.lstShowTheseY.ItemsSelected.Count
    'Debug.Print "ColumnLabelCount = " &
Me.lstShowTheseY.ItemsSelected.Count

    Dim row As Integer
    Dim col As Integer
    Dim iX As Integer
    Dim iY As Integer

    'Set column labels
    rst.MoveFirst
    For iY = LBound(aryItemsSelectedY) To
UBound(aryItemsSelectedY) - 1
        'Debug.Print
Me.chtTheGraph.DataGrid.ColumnLabel(iY + 1, 1) & " = "
& rst.Fields(aryItemsSelectedY(iY)).Name
        Me.chtTheGraph.DataGrid.ColumnLabel(iY + 1, 1) =
rst.Fields(aryItemsSelectedY(iY)).Name
        Next 'of aryItemsSelectedY()

    'Set row labels.
    For iX = LBound(aryItemsSelectedX) To
UBound(aryItemsSelectedX) - 1
        rst.MoveFirst
        For row = 0 To (rst.RecordCount - 1)
            If aryItemsSelectedX(iX) = row Then
                Me.chtTheGraph.DataGrid.RowLabel(iX + 1, 1) =
rst.Fields(0)
                'Debug.Print "Row: " & iX + 1 & " Label: " &
rst.Fields(0)
            End If
            rst.MoveNext
        Next

        'Load the data.
        rst.MoveFirst
        Dim nullflag As Boolean
        'Loop through all X values
        For row = 0 To (rst.RecordCount - 1)
            If aryItemsSelectedX(iX) = row Then
                For iY = LBound(aryItemsSelectedY) To
UBound(aryItemsSelectedY) - 1
                    'Debug.Print "Row: " & iX + 1 & ", Col: " &
iY + 1 & " Value: " & rst.Fields(aryItemsSelectedY(iY))
                    If IsNull(rst.Fields(aryItemsSelectedY(iY)))
Then nullflag = True
                    Me.chtTheGraph.DataGrid.SetData iX + 1, iY
+ 1, rst.Fields(aryItemsSelectedY(iY)), nullflag
                    nullflag = False
                Next
            End If
            rst.MoveNext
        Next
        Next 'of aryItemsSelectedX()

End Sub

'=====
'Function/Sub Name: Form_Close()
'

```

'Description: Closes the form.

'Input: None

'Output: None

'References: None

Private Sub Form\_Close()

Application.CommandBars("mnuOther").Visible = True  
Application.CommandBars("mnuPrintGraph").Visible = False

'Clean up  
rst.Close  
Set oCmd = Nothing  
cnn.Close

End Sub

Function/Sub Name: Form\_Activate()

'Description: Update the menu bar.

'Input: None

'Output: None

'References: None

Private Sub Form\_Activate()

Application.CommandBars("mnuPrintGraph").Visible = True  
End Sub

Function/Sub Name: Form\_Deactivate()

'Description: Updates the menu bar.

'Input : None

'Output: None

'References: None

Private Sub Form\_Deactivate()

Application.CommandBars("mnuPrintGraph").Visible = False  
End Sub

Function/Sub Name: Form\_Load()

'Description: Dynamically resizes the form to the users screen

'resolution and then centers it.

'Input: None

'Output: None

'References:

- ezSizeForm

Private Sub Form\_Load()

'Dynamically resize the form based on screen resolution.  
ezSizeForm Me, -1

MoveToCenter "4-0-1 -2-frm-TheActualGraph"

End Sub

Function/Sub Name: Form\_Open

'Description: Builds the MSChart20 control's Datagrid based upon

'the results of a RAC stored procedure (4-0-1 -0-  
flanCrossTabForGraphing).

'Also, sets up visual aspects of the graph and populates the X and

'Y multi-select listboxes with values.

'HINT: Uncomment the debug.print lines to troubleshoot this code.

'Input: None

'Output: None

'References: None

Private Sub Form\_Open(Cancel As Integer)

'Update menu bars.

Application.CommandBars("mnuOther").Visible = False

Application.CommandBars("mnuPrintGraph").Visible = True

'Set default button values.

Me.togEnlarge.Value = 0

Me.chkStack.Value = False

Me.chkTranspose.Value = False

'Run a stored procedure to get the graph data

Dim objPrmColleft As ADODB.Parameter

Dim objPrmColtop As ADODB.Parameter

Set cnn = CurrentProject.Connection

cnn.CursorLocation = adUseClient

Set rst = New ADODB.Recordset

Set oCmd = New ADODB.Command

oCmd.ActiveConnection = cnn

oCmd.CommandText = ""4-0-1 -0-

flanCrossTabForGraphing""

oCmd.CommandType = adCmdStoredProc

'Create parameters for the SP.

'They have to be appended in the same order that they appear in  
'the stored procedure.

Set objPrmColleft = oCmd.CreateParameter("@colleft",  
adVarChar, adParamInput, 500)

oCmd.Parameters.Append objPrmColleft

objPrmColleft.Value =

GlobalDeclarations.gStrXFieldToGraph

```

Set objPrmColtop = oCmd.CreateParameter("@coltop",
adVarChar, adParamInput, 500)
oCmd.Parameters.Append objPrmColtop
objPrmColtop.Value =
GlobalDeclarations.gStrYFieldToGraph

'Run the SP
Set rst = oCmd.Execute

'Build the data grid for the chart . . . this is how MSChart
objects
'get input.
Dim row As Integer
Dim col As Integer

Me.chtTheGraph.DataGrid.RowCount = rst.RecordCount
'Debug.Print "RowCount = " & rst.RecordCount
Me.chtTheGraph.DataGrid.RowLabelCount =
rst.RecordCount + 1
'Debug.Print "RowLabelCount = " & rst.RecordCount +
1
Me.chtTheGraph.DataGrid.ColumnCount =
rst.Fields.Count - 1
'Debug.Print "ColumnCount = " & rst.Fields.Count - 1
Me.chtTheGraph.DataGrid.ColumnLabelCount =
rst.Fields.Count - 1
'Debug.Print "ColumnLabelCount = " &
rst.Fields.Count - 1

'Debug.Print

'Set row labels
'First declare a temporary string to create values for the list
box.
Dim tempvaluelistX As String
Dim tempvaluelistY As String
tempvaluelistX = ""
tempvaluelistY = ""

rst.MoveFirst
For row = 0 To (rst.RecordCount - 1)
    Me.chtTheGraph.DataGrid.RowLabel(row + 1, 1) =
Trim(rst.Fields(0))
'Debug.Print "Row: " & row + 1 & " Label: " &
rst.Fields(0)
'Add the label to the list box.

'Replace commas and semicolons with dashes
becuase the mess up
'the list
Dim astrText As String
Dim iCount As Integer
astrText = Trim(rst.Fields(0))

'Loop through array, replacing commas and
semicolons
For iCount = 1 To Len(astrText)

    If Mid(astrText, iCount, 1) = "," Or
Mid(astrText, iCount, 1) = ";" Then
        'If array element satisfies wildcard search,
'replace it.
        Mid(astrText, iCount, 1) = "-"
    End If
Next

'Join string.

```

```

tempvaluelistX = tempvaluelistX & row & ";" &
astrText & ";"

'Debug.Print "Col1 " & row; " Col2: " & rst.Fields(0)
rst.MoveNext
Next

'Populate the Select X list box
Me.lstShowTheseX.ColumnCount = 2
Me.lstShowTheseX.ColumnWidths = "0;1"
Me.lstShowTheseX.RowSourceType = "Value List"
Me.lstShowTheseX.RowSource = tempvaluelistX
'Select all the values
ReDim aryItemsSelectedX(Me.lstShowTheseX.ListCount)
Dim iListItemIndex As Integer
For iListItemIndex = 0 To Me.lstShowTheseX.ListCount -
1
    Me.lstShowTheseX.Selected(iListItemIndex) = True
    aryItemsSelectedX(iListItemIndex) = iListItemIndex
Next

'Set column labels
'First column is already done.
'Other columns.
rst.MoveFirst
Dim j As Integer
j = 1
For col = 0 To (rst.Fields.Count - 2)
    Me.chtTheGraph.DataGrid.ColumnLabel(j, 1) =
Trim(rst.Fields(col + 1).Name)
'Debug.Print "Col: " & col + 1 & " Label: " &
rst.Fields((col + 1)).Name
'Add the label to the list box.

'Replace commas and semicolons with dashes
becuase the mess up
'the list
'Check for null fields and only operate on those that
are not null
astrText = Trim(rst.Fields(col + 1).Name)

' Loop through array, replacing commas and
semicolons
For iCount = 1 To Len(astrText)

    If Mid(astrText, iCount, 1) = "," Or
Mid(astrText, iCount, 1) = ";" Then
        'If array element satisfies wildcard search,
'replace it.
        Mid(astrText, iCount, 1) = "-"
    End If
Next
'Join string.
tempvaluelistY = tempvaluelistY & j & ";" &
astrText & ";"

'Debug.Print "Col1 " & j & " Col2: " & rst.Fields(col +
1).Name
j = j + 1
Next

'Populate the Select Y list box
Me.lstShowTheseY.ColumnCount = 2
Me.lstShowTheseY.ColumnWidths = "0;1"
Me.lstShowTheseY.RowSourceType = "Value List"
Me.lstShowTheseY.RowSource = tempvaluelistY
'Select all the values
ReDim aryItemsSelectedY(Me.lstShowTheseY.ListCount)

```



```

For iListItemIndex = 0 To Me.lstShowTheseY.ListCount -
1
    Me.lstShowTheseY.Selected(iListItemIndex) = True
    aryItemsSelectedY(iListItemIndex) = iListItemIndex +
1
Next

'Load the data.
rst.MoveFirst
Dim nullflag As Boolean
For row = 0 To (rst.RecordCount - 1)
    For col = 0 To (rst.Fields.Count - 2)
        'Debug.Print "Row: " & row + 1 & ", Col: " & col + 1
        & " Value: " & rst.Fields(col + 1)
        If IsNull(rst.Fields(col + 1)) Then nullflag = True
        Me.chtTheGraph.DataGrid.SetData row + 1, col + 1,
rst.Fields(col + 1), nullflag
        nullflag = False
    Next
    rst.MoveNext
Next

'Leave for future use.
'Use manual scale to display y axis (value axis)
'With
Me.chtTheGraph.Plot.Axis(VtChAxisIdY).ValueScale
'    .Auto = False
'    .Minimum = 0
'    .Maximum = Int(maxvalue * 1.1)
'End With

'Use manual scale to display x axis
'With
Me.chtTheGraph.Plot.Axis(VtChAxisIdX).ValueScale
'    .Auto = False
'    .Minimum = 0
'    .Maximum =
Me.chtTheGraph.DataGrid.RowLabelCount
'End With

'Set Font size for X Axis
Dim currentaxis As MSChart20Lib.Axis
Dim currentlabel As MSChart20Lib.Label
'Get a reference to the x axis
Set currentaxis =
Me.chtTheGraph.Plot.Axis(VtChAxisIdX)
'Loop though and set the font of each label
For Each currentlabel In currentaxis.Labels
    currentlabel.VtFont.Name = "small fonts"
    currentlabel.VtFont.Size = 7
Next currentlabel

'set up the legend
With Me.chtTheGraph
    .Legend.Location.LocationType =
VtChLocationTypeTop
    .Legend.VtFont.Style = VtFontStyleBold
    .Legend.Location.RECT.Max.Set 7560, 5132
    .Legend.Location.RECT.Min.Set 3004, 4864
End With
End Sub

```

---

```

'Function/Sub Name: fraChart_AfterUpdate()

```

```

'Description: Sets the ChartType option of the MSChart
control
'in response to a radio button selection. It has to check
'the value of fraDimensions to do this, so it knows if the
'chart should be 2d or 3d.

```

```

'Input : None
'
'Output: None
'
'References: fraDimensions.value

```

---

```

Private Sub fraChart_AfterUpdate()

```

```

    Select Case Me.fraChart.Value
    Case 1
        If Me.fraDimensions.Value = 1 Then
            Me.chtTheGraph.chartType =
VtChChartType2dBar
        Else
            Me.chtTheGraph.chartType =
VtChChartType3dBar
        End If
    Case 2
        If Me.fraDimensions.Value = 1 Then
            Me.chtTheGraph.chartType =
VtChChartType2dLine
        Else
            Me.chtTheGraph.chartType =
VtChChartType3dLine
        End If
    Case 3
        If Me.fraDimensions.Value = 1 Then
            Me.chtTheGraph.chartType =
VtChChartType2dArea
        Else
            Me.chtTheGraph.chartType =
VtChChartType3dArea
        End If
    Case 4
        If Me.fraDimensions.Value = 1 Then
            Me.chtTheGraph.chartType =
VtChChartType2dStep
        Else
            Me.chtTheGraph.chartType =
VtChChartType3dStep
        End If
    End Select

```

```

End Sub

```

---

```

'Function/Sub Name: fraDimensions_AfterUpdate()

```

```

'Description: Sets the ChartType option with respect to
number
'of dimensions (2d or 3d) of the MSChart control
'in response to a radio button selection. It has to check
'the value of fraChartType to do this, so it knows what style
'chart to create.

```

```

'Input: None
'
'Output: None

```

```

'References: fraChart.Value

```

```

'=====
Private Sub fraDimensions_AfterUpdate()

    If Me.fraDimensions.Value = 1 Then
        Select Case Me.fraChart.Value
            Case 1
                Me.chtTheGraph.chartType =
                VtChChartType2dBar
            Case 2
                Me.chtTheGraph.chartType =
                VtChChartType2dLine
            Case 3
                Me.chtTheGraph.chartType =
                VtChChartType2dArea
            Case 4
                Me.chtTheGraph.chartType =
                VtChChartType2dStep
            End Select
        Else
            Select Case Me.fraChart.Value
                Case 1
                    Me.chtTheGraph.chartType =
                    VtChChartType3dBar
                Case 2
                    Me.chtTheGraph.chartType =
                    VtChChartType3dLine
                Case 3
                    Me.chtTheGraph.chartType =
                    VtChChartType3dArea
                Case 4
                    Me.chtTheGraph.chartType =
                    VtChChartType3dStep
            End Select
            Me.lblTips.Caption = "Hold down the Ctrl key and
            mouse down to rotate the chart."
        End If
    End Sub

'=====
'Function/Sub Name: lstShowTheseX_AfterUpdate()
'
'Description: Builds the array used by cmdUpdate_Click() to
update
'the datagrid rows (X Axis) based on the users X-axis
selections.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub lstShowTheseX_AfterUpdate()

    Dim lst As ListBox
    Dim varItem As Variant
    Dim intIndex As Integer
    Dim intCount As Integer
    Dim intRow As Integer
    Dim intRows As Integer
    Dim intColumn As Integer
    Dim intColumns As Integer

    Set lst = Me.lstShowTheseX

```

```

'Check that at least one value has been selected
If lst.ItemsSelected.Count = 0 Then
    MsgBox "Please select at least one X value"
    Me.lblTips.Caption = "You must select at least one X-
Axis and one Y-Axis value."
    lst.SetFocus
    Me.cmdUpdate.Enabled = False 'disable update button.
    Exit Sub
End If

'Since an item was selected from the list box, enable the
update button
If Me.lstShowTheseY.ItemsSelected.Count >= 1 Then
    Me.lblTips.Caption = "Select a point to see its value."
    Me.cmdUpdate.Enabled = True
Else
    Me.lblTips.Caption = "You must select at least one Y-
Axis value, too."
End If

'Get the count of selected items and redim the array to hold
them
intColumns = lst.ColumnCount
intRows = lst.ItemsSelected.Count
ReDim aryItemsSelectedX(intRows)

'Add the index of the value selected in the box to the array.
This
'index corresponds to the index of the value in the
recordset that
'was queried when the form opened. This will be used
when the user
'clicks the Update button to select just those records.
Dim i As Integer
i = 0
For Each varItem In lst.ItemsSelected
    aryItemsSelectedX(i) = Nz(lst.Column(0, varItem))
    'Debug.Print "Added to Array: " & aryItemsSelectedX(i)
    i = i + 1
Next varItem

'This code prints the output to the debug window.
Uncomment to
'help debug.
'Debug.Print "-----"
'Dim s As String
's = ""
'For i = LBound(aryItemsSelectedX) To
UBound(aryItemsSelectedX) - 1
'    s = s & aryItemsSelectedX(i) & ", "
'Next
'Debug.Print s

End Sub

```

```

Private Sub lstShowTheseX_LostFocus()
    Me.lblTips.Caption = "Select a point to see its value."
End Sub

'=====
'Function/Sub Name: lstShowTheseY_AfterUpdate()
'
'Description: Builds the array used by cmdUpdate_Click() to
update

```

```

'the datagrid columns (Y Axis) based on the users Y-axis
selections.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub lstShowTheseY_AfterUpdate()

    Dim lst As ListBox
    Dim varItem As Variant
    Dim intIndex As Integer
    Dim intCount As Integer
    Dim intRow As Integer
    Dim intRows As Integer
    Dim intColumn As Integer
    Dim intColumns As Integer

    Set lst = Me.lstShowTheseY

    'Check that at least one value has been selected
    If lst.ItemsSelected.Count = 0 Then
        MsgBox "Please select at least one Y value"
        Me.lblTips.Caption = "You must select at least one X-
Axis and one Y-Axis value."
        lst.SetFocus
        Me.cmdUpdate.Enabled = False 'disable update button.
        Exit Sub
    End If

    'Since an item was selected from the list box, enable the
update button
    If Me.lstShowTheseX.ItemsSelected.Count >= 1 Then
        Me.lblTips.Caption = "Select a point to see its value."
        Me.cmdUpdate.Enabled = True
    Else
        Me.lblTips.Caption = "You must select at least one X-
Axis value, too."
    End If

    'Get the count of selected items and redim the array to hold
them
    intColumns = lst.ColumnCount
    intRows = lst.ItemsSelected.Count
    ReDim aryItemsSelectedY(intRows)

    'Add the index of the value selected in the box to the array.
This
    'index corresponds to the index of the value in the
recordset that
    'was queried when the form opened. This will be used
when the user
    'clicks the Update button to select just those records.
    Dim i As Integer
    i = 0
    For Each varItem In lst.ItemsSelected
        aryItemsSelectedY(i) = Nz(lst.Column(0, varItem))
        Debug.Print "Added to Array: " & aryItemsSelectedY(i)
        i = i + 1
    Next varItem

    'This code prints the output to the debug window.
Uncomment to
    'help debug.
    Debug.Print "-----"

```

```

Dim s As String
's = ""
For i = LBound(aryItemsSelectedY) To
UBound(aryItemsSelectedY) - 1
    s = s & aryItemsSelectedY(i) & ", "
Next
Debug.Print s

End Sub

=====
'Function/Sub Name: lstShowTheseY_LostFocus()
'
'Description: Updates the "Tips" label with information for
the
'user.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub lstShowTheseY_LostFocus()
    Me.lblTips.Caption = "Select a point to see its value."
End Sub

=====
'Function/Sub Name: Option13_LostFocus()
'
'Description: Updates the "Tips" label with information for
the
'user.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Option13_LostFocus()
    Me.lblTips.Caption = "Select a point to see its value."
End Sub

=====
'Function/Sub Name: togEnlarge_AfterUpdate()
'
'Description: Enlarges or shrinks the form using the
ezSizeMode
'class.
'
'Input: None
'
'Output: None
'
'References: ezSizeMode
'
=====
Private Sub togEnlarge_AfterUpdate()

    'Make it big
    If Me.togEnlarge.Value = -1 Then
        ezSizeMode Me, 1.37
    End If

```

```

        Me.ScrollBars = 3
        DoCmd.Maximize

'Make it small
Else
    DoCmd.Restore
    Me.ScrollBars = 0
    ezSizeForm Me, 0.73
    Me.Repaint
End If

End Sub

'=====
'Function/Sub Name: chtTheGraph_PointSelected
'
'Description: Updates the "Tips" label with information
'specified
'when the user clicks on a datapoint in the MSChart20 object.
'
'Input: Automatically generated by a mouse click.
'
'Output: None
'
'References: None
'
'=====
Private Sub chtTheGraph_PointSelected(Series As Integer, _
    DataPoint As Integer, MouseFlags As Integer, Cancel As
Integer)
    'This allows the user to see the value of any particular data
    point in a
    'series by selecting it. The value of the data point is shown
    in the label
    'named lblTips.

    If Me.chkTranspose.Value = False Then
        Me.chtTheGraph.Column = Series
        Me.chtTheGraph.row = DataPoint
        Me.lblTips.Caption = "Value of Series " & Chr(34) &
        Me.chtTheGraph.DataGrid.ColumnLabel(Series, 1) &
        Chr(34) & ", point " & Chr(34) &
        Me.chtTheGraph.DataGrid.RowLabel(DataPoint, 1) &
        Chr(34) & " = " & Me.chtTheGraph.Data
    Else
        Me.chtTheGraph.Column = DataPoint
        Me.chtTheGraph.row = Series
        Me.lblTips.Caption = "Value of Series " & Chr(34) &
        Me.chtTheGraph.DataGrid.RowLabel(Series, 1) & Chr(34)
        & ", point " & Chr(34) &
        Me.chtTheGraph.DataGrid.ColumnLabel(DataPoint, 1) &
        Chr(34) & " = " & Me.chtTheGraph.Data
    End If
End Sub

'=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'
'    - clFormWindow
'
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        .Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing
End Sub

```

## **FORMCLASS-7-0-0-1-PopUpFrm-waitProgressBar**

```
Option Compare Database
Option Explicit
'#####
'      FORM DESCRIPTION
'#####
'Class Name: 7-0-0-1-PopUpFrm-waitProgressBar
'
'Author: Pat Flanders & Scott Tufts
'
This class shows a pop-up form with spinning globe while
data
for other forms is being loaded.
'
References:
'      - clFormWindow
'      - ez_SizingFunctions
'      - ConnectionFunctions
'#####

*****
'      FUNCTIONS
*****

'=====
'Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'
References:
'      - ezSizeForm
'
'=====
Private Sub Form_Load()
    ezSizeForm Me, -1
    MoveToCenter "7-0-0-1-PopUpFrm-waitProgressBar"
End Sub

'=====
'Function/Sub Name: Form_Close()
'
'Description: Ensures the mouspointer gets set back to
normal.
'
'Input: None
'
'Output: None
'
References: None
'
'=====
Private Sub Form_Close()
    Screen.MousePointer = 0
End Sub

'=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
basis.
'
'Input: None
'
'Output: None
'
References:
'      - clFormWindow
'
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        .Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

End Sub
```

## **FORMCLASS-8-0-0-1-fm-Reports**

Option Compare Database

Option Explicit

'#####

### FORM DESCRIPTION

'#####

'Class Name: 8-0-0-1-fm-Reports

'

'Author: Pat Flanders & Scott Tufts

'

'This class is the form for selecting the type of report to run.

'

'References:

- ' - clFormWindow
- ' - ez\_SizingFunctions
- ' - GlobalDeclarations
- ' - All reports

'

'#####

\*\*\*\*\*

### FUNCTIONS

\*\*\*\*\*

'=====

'Function/Sub Name: cmdCloseReportMenu\_Click()

'

'Description: Closes the form.

'

'Input: None

'

'Output: None

'

'References: None

'

'=====

Private Sub cmdCloseReportMenu\_Click()

DoCmd.Close acForm, "8-0-0-1-fm-Reports"

End Sub

'=====

'Function/Sub Name:

- ' - cmdAll\_Click()
- ' - cmdAircraft\_Click()
- ' - cmdClass\_Click()
- ' - cmdLocation\_Click()
- ' - cmdOrganization\_Click()
- ' - cmdType\_Click()
- ' - cmdYear\_Click()
- ' - cmdChron\_Click()
- ' - cmdCloseReportMenu\_Click()

'

'Description: The following 9 functions launch their respective

'reports in response to command button click events.

'

'Input: None

'

'Output: None

'

'References: None

'

'=====

Private Sub cmdAircraft\_Click()

On Error GoTo startError

Me.Visible = False

ConnectionFunctions.waitScreen "2-2-Distribution-Aircraft"

DoCmd.OpenReport "2-2-Distribution-Aircraft",

acViewPreview

exitSub:

Exit Sub

startError:

MsgBox "You must have a default printer installed in order to preview reports.", vbCritical + vbOKOnly, "Can't Find A Printer"

End Sub

Private Sub cmdAll\_Click()

On Error GoTo startError

Me.Visible = False

ConnectionFunctions.waitScreen "2-1-Distribution-AllMishaps"

DoCmd.OpenReport "2-1-Distribution-AllMishaps",

acViewPreview

exitSub:

Exit Sub

startError:

MsgBox "You must have a default printer installed in order to preview reports.", vbCritical + vbOKOnly, "Can't Find A Printer"

End Sub

Private Sub cmdClass\_Click()

On Error GoTo startError

Me.Visible = False

ConnectionFunctions.waitScreen "2-4-Distribution-Class"

DoCmd.OpenReport "2-4-Distribution-Class",

acViewPreview

exitSub:

Exit Sub

startError:

MsgBox "You must have a default printer installed in order to preview reports.", vbCritical + vbOKOnly, "Can't Find A Printer"

End Sub

Private Sub cmdLocation\_Click()

On Error GoTo startError

Me.Visible = False

ConnectionFunctions.waitScreen "2-3-Distribution-Location"

DoCmd.OpenReport "2-3-Distribution-Location",

acViewPreview

exitSub:

Exit Sub

startError:

MsgBox "You must have a default printer installed in order to preview reports.", vbCritical + vbOKOnly, "Can't Find A Printer"

End Sub

Private Sub cmdOrganization\_Click()

On Error GoTo startError

Me.Visible = False

ConnectionFunctions.waitScreen "2-5-Distribution-Organization"

```

    DoCmd.OpenReport "2-5-Distribution-Organization",
acViewPreview
exitSub:
    Exit Sub
startError:
    MsgBox "You must have a default printer installed in
order to preview reports.", vbCritical + vbOKOnly, "Can't
Find A Printer"
End Sub
Private Sub cmdYear_Click()
    On Error GoTo startError
    Me.Visible = False
    ConnectionFunctions.waitScreen "2-7-Distribution-Year"
    DoCmd.OpenReport "2-7-Distribution-Year",
acViewPreview
exitSub:
    Exit Sub
startError:
    MsgBox "You must have a default printer installed in
order to preview reports.", vbCritical + vbOKOnly, "Can't
Find A Printer"
End Sub
Private Sub cmdType_Click()
    On Error GoTo startError
    Me.Visible = False
    ConnectionFunctions.waitScreen "2-6-Distribution-Type"
    DoCmd.OpenReport "2-6-Distribution-Type",
acViewPreview
exitSub:
    Exit Sub
startError:
    MsgBox "You must have a default printer installed in
order to preview reports.", vbCritical + vbOKOnly, "Can't
Find A Printer"
End Sub
Private Sub cmdChron_Click()
    On Error GoTo startError
    Me.Visible = False
    ConnectionFunctions.waitScreen "3-0-Chronological-
AllMishaps"
    DoCmd.OpenReport "3-0-Chronological-AllMishaps",
acViewPreview
exitSub:
    Exit Sub
startError:
    MsgBox "You must have a default printer installed in
order to preview reports.", vbCritical + vbOKOnly, "Can't
Find A Printer"
End Sub

```

---

Function/Sub Name:

```

'
'   - cmdAll_MouseMove()
'   - cmdAircraft_MouseMove()
'   - cmdClass_MouseMove()
'   - cmdLocation_MouseMove()
'   - cmdOrganization_MouseMove()
'   - cmdType_MouseMove()
'   - cmdYear_MouseMove()
'   - cmdChron_MouseMove()
'   - cmdCloseReportMenu_MouseMove()
'

```

Description: The following 9 functions update text color on the command buttons in response to mouse over events.

Input: None

```

'
'Output: None
'
References: None
'

```

---

```

Private Sub cmdAll_MouseMove(Button As Integer, Shift As
Integer, X As Single, Y As Single)
' Make the button text blue when it gets the focus
Me.cmdAll.ForeColor = QBColor(9)
Me.cmdAircraft.ForeColor = QBColor(0)
Me.cmdClass.ForeColor = QBColor(0)
Me.cmdLocation.ForeColor = QBColor(0)
Me.cmdOrganization.ForeColor = QBColor(0)
Me.cmdType.ForeColor = QBColor(0)
Me.cmdChron.ForeColor = QBColor(0)
Me.cmdCloseReportMenu.ForeColor = QBColor(0)
Me.cmdYear.ForeColor = QBColor(0)
End Sub

```

```

Private Sub cmdAircraft_MouseMove(Button As Integer,
Shift As Integer, X As Single, Y As Single)
' Make the button text blue when it gets the focus
Me.cmdAll.ForeColor = QBColor(0)
Me.cmdAircraft.ForeColor = QBColor(9)
Me.cmdClass.ForeColor = QBColor(0)
Me.cmdLocation.ForeColor = QBColor(0)
Me.cmdOrganization.ForeColor = QBColor(0)
Me.cmdType.ForeColor = QBColor(0)
Me.cmdChron.ForeColor = QBColor(0)
Me.cmdCloseReportMenu.ForeColor = QBColor(0)
Me.cmdYear.ForeColor = QBColor(0)
End Sub

```

```

Private Sub cmdClass_MouseMove(Button As Integer, Shift
As Integer, X As Single, Y As Single)
' Make the button text blue when it gets the focus
Me.cmdAll.ForeColor = QBColor(0)
Me.cmdAircraft.ForeColor = QBColor(0)
Me.cmdClass.ForeColor = QBColor(9)
Me.cmdLocation.ForeColor = QBColor(0)
Me.cmdOrganization.ForeColor = QBColor(0)
Me.cmdType.ForeColor = QBColor(0)
Me.cmdChron.ForeColor = QBColor(0)
Me.cmdCloseReportMenu.ForeColor = QBColor(0)
Me.cmdYear.ForeColor = QBColor(0)
End Sub

```

```

Private Sub cmdLocation_MouseMove(Button As Integer,
Shift As Integer, X As Single, Y As Single)
' Make the button text blue when it gets the focus
Me.cmdAll.ForeColor = QBColor(0)
Me.cmdAircraft.ForeColor = QBColor(0)
Me.cmdClass.ForeColor = QBColor(0)
Me.cmdLocation.ForeColor = QBColor(9)
Me.cmdOrganization.ForeColor = QBColor(0)
Me.cmdType.ForeColor = QBColor(0)
Me.cmdChron.ForeColor = QBColor(0)
Me.cmdCloseReportMenu.ForeColor = QBColor(0)
Me.cmdYear.ForeColor = QBColor(0)
End Sub

```

```

Private Sub cmdOrganization_MouseMove(Button As
Integer, Shift As Integer, X As Single, Y As Single)
' Make the button text blue when it gets the focus
Me.cmdAll.ForeColor = QBColor(0)
Me.cmdAircraft.ForeColor = QBColor(0)
Me.cmdClass.ForeColor = QBColor(0)
Me.cmdLocation.ForeColor = QBColor(0)
Me.cmdOrganization.ForeColor = QBColor(9)
Me.cmdType.ForeColor = QBColor(0)

```

```

        Me.cmdYear.ForeColor = QBColor(0)
        Me.cmdChron.ForeColor = QBColor(0)
        Me.cmdCloseReportMenu.ForeColor = QBColor(0)
    End Sub
    Private Sub cmdType_MouseMove(Button As Integer, Shift
    As Integer, X As Single, Y As Single)
        ' Make the button text blue when it gets the focus
        Me.cmdAll.ForeColor = QBColor(0)
        Me.cmdAircraft.ForeColor = QBColor(0)
        Me.cmdClass.ForeColor = QBColor(0)
        Me.cmdLocation.ForeColor = QBColor(0)
        Me.cmdOrganization.ForeColor = QBColor(0)
        Me.cmdType.ForeColor = QBColor(9)
        Me.cmdChron.ForeColor = QBColor(0)
        Me.cmdCloseReportMenu.ForeColor = QBColor(0)
        Me.cmdYear.ForeColor = QBColor(0)
    End Sub
    Private Sub cmdChron_MouseMove(Button As Integer, Shift
    As Integer, X As Single, Y As Single)
        ' Make the button text blue when it gets the focus
        Me.cmdAll.ForeColor = QBColor(0)
        Me.cmdAircraft.ForeColor = QBColor(0)
        Me.cmdClass.ForeColor = QBColor(0)
        Me.cmdLocation.ForeColor = QBColor(0)
        Me.cmdOrganization.ForeColor = QBColor(0)
        Me.cmdType.ForeColor = QBColor(0)
        Me.cmdChron.ForeColor = QBColor(9)
        Me.cmdCloseReportMenu.ForeColor = QBColor(0)
        Me.cmdYear.ForeColor = QBColor(0)
    End Sub
    Private Sub cmdCloseReportMenu_MouseMove(Button As
    Integer, Shift As Integer, X As Single, Y As Single)
        ' Make the button text blue when it gets the focus
        Me.cmdAll.ForeColor = QBColor(0)
        Me.cmdAircraft.ForeColor = QBColor(0)
        Me.cmdClass.ForeColor = QBColor(0)
        Me.cmdLocation.ForeColor = QBColor(0)
        Me.cmdOrganization.ForeColor = QBColor(0)
        Me.cmdType.ForeColor = QBColor(0)
        Me.cmdChron.ForeColor = QBColor(0)
        Me.cmdCloseReportMenu.ForeColor = QBColor(9)
        Me.cmdYear.ForeColor = QBColor(0)
    End Sub
    Private Sub cmdYear_MouseMove(Button As Integer, Shift
    As Integer, X As Single, Y As Single)
        ' Make the button text blue when it gets the focus
        Me.cmdAll.ForeColor = QBColor(0)
        Me.cmdAircraft.ForeColor = QBColor(0)
        Me.cmdClass.ForeColor = QBColor(0)
        Me.cmdLocation.ForeColor = QBColor(0)
        Me.cmdOrganization.ForeColor = QBColor(0)
        Me.cmdType.ForeColor = QBColor(0)
        Me.cmdYear.ForeColor = QBColor(9)
        Me.cmdChron.ForeColor = QBColor(0)
        Me.cmdCloseReportMenu.ForeColor = QBColor(0)
    End Sub

```

```

=====
Function/Sub Name: Form_Close()
Description: Closes the form.
Input: None
Output: None
References: None

```

```

Private Sub Form_Close()
    Application.CommandBars("mnuOther").Visible = False
    Application.CommandBars("mnuProgramMain").Visible =
    True
    Forms![MainMenu].Visible = True
End Sub

=====
Function/Sub Name: Form_Activate()
Description: Update the menu bar.
Input: None
Output: None
References: None
Private Sub Form_Activate()
    Application.CommandBars("mnuOther").Visible = True
End Sub

=====
Function/Sub Name: Form_Deactivate()
Description: Updates the menu bar.
Input: None
Output: None
References: None
Private Sub Form_Deactivate()
    Application.CommandBars("mnuOther").Visible = False
End Sub

=====
Function/Sub Name: Form_Load()
Description: Dynamically resizes the form to the users
screen
resolution and then centers it.
Input: None
Output: None
References:
    - ezSizeMode
Private Sub Form_Load()
    ezSizeMode Me, -1
    MoveToCenter "8-0-0-1-frm-Reports"
End Sub

```



```

=====
'Function/Sub Name: Form_Open
'
'Description: Updates the menu bar and sets the focus to the
first
'command button, setting its color to blue.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Form_Open(Cancel As Integer)

    Forms![MainMenu].Visible = False
    Application.CommandBars("mnuOther").Visible = True

    Me.cmdCloseReportMenu.SetFocus

    'Make the button text blue when it gets the focus
    Me.cmdAll.ForeColor = QBColor(0)
    Me.cmdAircraft.ForeColor = QBColor(0)
    Me.cmdClass.ForeColor = QBColor(0)
    Me.cmdLocation.ForeColor = QBColor(0)
    Me.cmdOrganization.ForeColor = QBColor(0)
    Me.cmdType.ForeColor = QBColor(0)
    Me.cmdChron.ForeColor = QBColor(0)
    Me.cmdCloseReportMenu.ForeColor = QBColor(0)
    Me.cmdYear.ForeColor = QBColor(0)

End Sub

```

```

=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeMode
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'         - clFormWindow
'
=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        '.Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

End Sub

```

## **FORMCLASS-MainMenu**

```

Option Compare Database
Option Explicit
#####
'          FORM DESCRIPTION
#####
'Class Name: MainMenu
'
'Author: Pat Flanders & Scott Tufts
'
'Description: This class is the main switchboard for the
program.
'It is responsible for launching all other processes, connecting
to the SQL server, validating Administrator settings, and
determining
'O/S platform.
'
'References:
'   - Connection functions
'   - clFormWindow
'   - ez_SizingFunctions
'   - GlobalDeclarations
'   - Numerous forms
'
#####

*****
'          FUNCTIONS
*****

=====
Function/Sub Name: Form_Activate()
'
'Description: Update the menu bar.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Form_Activate()
    Application.CommandBars("mnuProgramMain").Visible =
True
End Sub

=====
Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'
'References:
'   - ezSizeForm

```

```

'
=====
Private Sub Form_Load()

    Screen.MousePointer = 11

    DoEvents

    'Set the picture for military or civilian.
    If ez_SizingFunctions.ezGetScreenRes = "640x480" Or _
ez_SizingFunctions.ezGetScreenRes = "800x600" Or _
ez_SizingFunctions.ezGetScreenRes = "1024x768"
Then
        If GlobalDeclarations.gStrTypeDB = "M" Then
            Me.imgCivilian.Visible = False
            Me.imgMilitary.Visible = True
        ElseIf GlobalDeclarations.gStrTypeDB = "C" Then
            Me.imgMilitary.Visible = False
            Me.imgCivilian.Visible = True
        End If
    Else
        'Dynamically resize the form based on screen resolution.
        ezSizeForm Me, -1
        MoveToCenter "MainMenu"
        If GlobalDeclarations.gStrTypeDB = "M" Then
            Me.imgCivilian.Visible = False
            Me.imgMilitary.Visible = True
        ElseIf GlobalDeclarations.gStrTypeDB = "C" Then
            Me.imgMilitary.Visible = False
            Me.imgCivilian.Visible = True
        End If
    End If

    Screen.MousePointer = 0

End Sub

'
=====
Function/Sub Name: Form_Open()
'
'Description: Set initial screen colors, determine OS type, and
initiate
'connection to the SQL Server.
'
'Input: None
'
'Output: None
'
'References:
'   - ezSizeForm
'   - DetermineOSDeclares
'   - ConnectionFunctions
'
=====
Private Sub Form_Open(Cancel As Integer)

    'Check to see if the user accidentally opened a second
instance of HFACS
    'by mistake.
    If DetermineOSDeclares.IsRunning = -1 Then
        MsgBox "You can only run one instance of HFACS-ME
at a time. This instance will now close.", vbOKOnly +
vbExclamation, "HFACS Is Already Running"
    End If

```

```

        Screen.MousePointer = 11
        ConnectionFunctions.removeConnection
    End If

    Screen.MousePointer = 11

    ' Change menu color when mouse is over button
    Me.lblQuery.BackColor = QBColor(9) ' Blue
    Me.lblGraph.BackColor = QBColor(15) ' White
    Me.lblReport.BackColor = QBColor(15) ' White
    Me.lblAddEditMishaps.BackColor = QBColor(15) ' White
    Me.lblInvestigate.BackColor = QBColor(15) ' White
    Me.lblExit.BackColor = QBColor(15) ' White
    '
    Me.lblQuery.SpecialEffect = 1 ' Raised
    Me.lblGraph.SpecialEffect = 0 ' Normal
    Me.lblReport.SpecialEffect = 0 ' Normal
    Me.lblAddEditMishaps.SpecialEffect = 0 ' Normal
    Me.lblInvestigate.SpecialEffect = 0 ' Normal
    Me.lblExit.SpecialEffect = 0 ' Normal
    '
    Me.lblQuery.ForeColor = QBColor(15) ' White
    Me.lblGraph.ForeColor = QBColor(0) ' Black
    Me.lblReport.ForeColor = QBColor(0) ' Black
    Me.lblAddEditMishaps.ForeColor = QBColor(0) ' Black
    Me.lblInvestigate.ForeColor = QBColor(0) ' Black
    Me.lblExit.ForeColor = QBColor(0) ' Black

    'CreateConnection
    ConnectionFunctions.InitConnection
    lblServerConnectedTo.Caption = "Connected To Server: "
    & GlobalDeclarations.gStrServerName
    lblServerConnectedTo.Visible = True
    Screen.MousePointer = 0

End Sub

'=====
'Function/Sub Name: lblAddEditMishaps_Click()
'
'Description: Only Administrators can access the
administration
'functions and then, only for the local machine. This function
'ensures that the user is a Window O/S Administrator, a SQL
Server
'Administrator, and an HFACS Administrator. If all these
tests are
'passed, then the the 1-0-0-0-frm-SelectMishap form is
opened.
'
'Input: None
'
'Output: None
'
'References:
'
'    - Invesigate.mdb
'    - 1-0-0-6-PopUpFrm-AdministratorLogon
'    - 1-0-0-0-frm-SelectMishap
'
'=====
Private Sub lblAddEditMishaps_Click()

    On Error GoTo startError

    *** Step 1
    'Check to make sure the user is logged onto the local SQL
server.

    If GlobalDeclarations.gStrServerName = "(local)" Then

        *** Step 2
        'Now check to see if the user is a SQL Server sysadmin
by
        'passing a parameter to a stored procedure.
        Dim cnn As Connection
        Dim oCmd As ADODB.Command
        Dim rst As ADODB.Recordset
        Set cnn = CurrentProject.Connection
        cnn.CursorLocation = adUseClient
        Set rst = New ADODB.Recordset
        Set oCmd = New ADODB.Command
        oCmd.ActiveConnection = cnn
        oCmd.CommandText = """"1-0-0-4-
flanIsUserSysadmin""""
        oCmd.CommandType = adCmdStoredProc
        DoCmd.SetWarnings (False)
        Set rst = oCmd.Execute
        DoCmd.SetWarnings (True)
        rst.MoveFirst

        'Check for SQL SYSADMIN permissions
        If rst!IsUserOwner <> 1 Then
            MsgBox "You must have SQL SERVER
SYSADMIN permissions to administer the HFACS
database.", vbOKOnly + vbExclamation, "Insufficient
Permissions"
            GoTo exitSub
        End If

        *** Step 3
        'Check to make sure that the user is a Windows System
Administrator
        If Trim(oHFACSConnection.getServerPath) = ""
Then
            MsgBox "You must have Windows System
Administrator permissions to Administer HFACS.",
vbOKOnly + vbExclamation, "Insufficient Permissions"
            GoTo exitSub
        End If

        *** Step 4
        'Check to see if the user has already logged on as local
administrator
        'by checking the gBlnAdministrator flag, otherwise
prompt now.
        If GlobalDeclarations.gBlnAdministrator = True Then
            DoCmd.OpenForm "1-0-0-0-frm-SelectMishap"
        Else
            DoCmd.OpenForm "1-0-0-6-PopUpFrm-
AdministratorLogon"
        End If
        Else
            MsgBox "You can only administer the database when
logged onto the '(local)' server.", vbOKOnly +
vbExclamation, "Not Logged On To (local)"
        End If

    exitSub:
        On Error GoTo 0
        On Error Resume Next
        rst.Close
        Set oCmd = Nothing
        cnn.Close

    Exit Sub

startError:

```

```

startError:
    MsgBox ERR.Description & "Error number: " &
ERR.Number
    GoTo exitSub

End Sub
'=====
'Function/Sub Name: lblAddEditMishaps_MouseMove()
'
'Description: Sets command button text colors.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub lblAddEditMishaps_MouseMove(Button As
Integer, Shift As Integer, X As Single, Y As Single)

    ' Change menu color when mouse is over button
    Me.lblQuery.BackColor = QBColor(15) ' White
    Me.lblGraph.BackColor = QBColor(15) ' White
    Me.lblReport.BackColor = QBColor(15) ' White
    Me.lblAddEditMishaps.BackColor = QBColor(9) ' Blue
    Me.lblInvestigate.BackColor = QBColor(15) ' White
    Me.lblExit.BackColor = QBColor(15) ' White
    '
    Me.lblQuery.SpecialEffect = 0 ' Normal
    Me.lblGraph.SpecialEffect = 0 ' Normal
    Me.lblReport.SpecialEffect = 0 ' Normal
    Me.lblAddEditMishaps.SpecialEffect = 1 ' Raised
    Me.lblInvestigate.SpecialEffect = 0 ' Normal
    Me.lblExit.SpecialEffect = 0 ' Normal
    '
    Me.lblQuery.ForeColor = QBColor(0) ' Black
    Me.lblGraph.ForeColor = QBColor(0) ' Black
    Me.lblReport.ForeColor = QBColor(0) ' Black
    Me.lblAddEditMishaps.ForeColor = QBColor(15) ' White
    Me.lblInvestigate.ForeColor = QBColor(0) ' Black
    Me.lblExit.ForeColor = QBColor(0) ' Black

End Sub

'=====
'Function/Sub Name: lblExit_Click()
'
'Description: Closes the program and properly disconnects
from the
'SQL server.
'
'Input: None
'
'Output: None
'
'References:
'
'    - ConnectionFunctions
'
'=====
Private Sub lblExit_Click()

    ' Prompt to see if the user really wants to quit
    DoCmd.Beep
    Dim response As Variant

```

```

        response = MsgBox("Are you sure you want to Exit?",
vbYesNo + vbCritical + vbDefaultButton2, "Exit To
Windows?")
        If response = vbYes Then ' User chose Yes.
            ConnectionFunctions.removeConnection
        End If

End Sub

'=====
'Function/Sub Name: lblExit_MouseMove
'
'Description: Sets command button text colors.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub lblExit_MouseMove(Button As Integer, Shift As
Integer, X As Single, Y As Single)

    ' Change menu color when mouse is over button
    Me.lblQuery.BackColor = QBColor(15) ' White
    Me.lblGraph.BackColor = QBColor(15) ' White
    Me.lblReport.BackColor = QBColor(15) ' White
    Me.lblAddEditMishaps.BackColor = QBColor(15) ' White
    Me.lblInvestigate.BackColor = QBColor(15) ' White
    Me.lblExit.BackColor = QBColor(9) ' Blue
    '
    Me.lblQuery.SpecialEffect = 0 ' Normal
    Me.lblGraph.SpecialEffect = 0 ' Normal
    Me.lblReport.SpecialEffect = 0 ' Normal
    Me.lblAddEditMishaps.SpecialEffect = 0 ' Normal
    Me.lblInvestigate.SpecialEffect = 0 ' Normal
    Me.lblExit.SpecialEffect = 1 ' Raised
    '
    Me.lblQuery.ForeColor = QBColor(0) ' Black
    Me.lblGraph.ForeColor = QBColor(0) ' Black
    Me.lblReport.ForeColor = QBColor(0) ' Black
    Me.lblAddEditMishaps.ForeColor = QBColor(0) ' Black
    Me.lblInvestigate.ForeColor = QBColor(0) ' Black
    Me.lblExit.ForeColor = QBColor(15) ' White

End Sub

'=====
'Function/Sub Name: lblGraph_Click()
'
'Description: Opens the Expert graph form (4-0-1-0-fm-
ExpertGraph).
'
'Input: None
'
'Output: None
'
'References:
'
'    - 4-0-1-0-fm-ExpertGraph
'
'=====
Private Sub lblGraph_Click()

    DoCmd.OpenForm "4-0-1-0-fm-ExpertGraph"

```

End Sub

```
=====
Function/Sub Name: lblGraph_MouseMove
Description: Sets command button text colors.
Input: None
Output: None
References: None
Private Sub lblGraph_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Change menu color when mouse is over button
    Me.lblQuery.BackColor = QBColor(15) ' White
    Me.lblGraph.BackColor = QBColor(9) ' Blue
    Me.lblReport.BackColor = QBColor(15) ' White
    Me.lblAddEditMishaps.BackColor = QBColor(15) ' White
    Me.lblInvestigate.BackColor = QBColor(15) ' White
    Me.lblExit.BackColor = QBColor(15) ' White
    Me.lblQuery.SpecialEffect = 0 ' Normal
    Me.lblGraph.SpecialEffect = 1 ' Raised
    Me.lblReport.SpecialEffect = 0 ' Normal
    Me.lblAddEditMishaps.SpecialEffect = 0 ' Normal
    Me.lblInvestigate.SpecialEffect = 0 ' Normal
    Me.lblExit.SpecialEffect = 0 ' Normal
    Me.lblQuery.ForeColor = QBColor(0) ' Black
    Me.lblGraph.ForeColor = QBColor(15) ' White
    Me.lblReport.ForeColor = QBColor(0) ' Black
    Me.lblAddEditMishaps.ForeColor = QBColor(0) ' Black
    Me.lblInvestigate.ForeColor = QBColor(0) ' Black
    Me.lblExit.ForeColor = QBColor(0) ' Black
End Sub
```

```
=====
Function/Sub Name: lblInvestigate_Click()
Description: Launches the Invetigate.mdb Access database in a separate process.
Input: None
Output: None
References:
    - Investigate.mdb
Private Sub lblInvestigate_Click()
```

```
    Dim RetVal
    RetVal = Shell("MSACCESS.EXE " & Chr(34) & GlobalDeclarations.gStrAppPath & "Investigate.mdb" & Chr(34), 1) ' Run Access.
```

End Sub

```
=====
Function/Sub Name: lblInvestigate_MouseMove()
Description: Sets command button text colors.
Input: None
Output: None
References: None
Private Sub lblInvestigate_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Change menu color when mouse is over button
    Me.lblQuery.BackColor = QBColor(15) ' White
    Me.lblGraph.BackColor = QBColor(15) ' White
    Me.lblReport.BackColor = QBColor(15) ' White
    Me.lblAddEditMishaps.BackColor = QBColor(15) ' White
    Me.lblInvestigate.BackColor = QBColor(9) ' Blue
    Me.lblExit.BackColor = QBColor(15) ' White
    Me.lblQuery.SpecialEffect = 0 ' Normal
    Me.lblGraph.SpecialEffect = 0 ' Normal
    Me.lblReport.SpecialEffect = 0 ' Normal
    Me.lblAddEditMishaps.SpecialEffect = 0 ' Normal
    Me.lblInvestigate.SpecialEffect = 1 ' Raised
    Me.lblExit.SpecialEffect = 0 ' Normal
    Me.lblQuery.ForeColor = QBColor(0) ' Black
    Me.lblGraph.ForeColor = QBColor(0) ' Black
    Me.lblReport.ForeColor = QBColor(0) ' Black
    Me.lblAddEditMishaps.ForeColor = QBColor(0) ' Black
    Me.lblInvestigate.ForeColor = QBColor(15) ' White
    Me.lblExit.ForeColor = QBColor(0) ' Black
End Sub
```

```
=====
Function/Sub Name: lblQuery_Click()
Description: Opens the Expert graph form (2-0-1-0-fm-QueryMenu).
Input: None
Output: None
References:
    - 2-0-1-0-fm-QueryMenu
Private Sub lblQuery_Click()
```

```
    DoCmd.OpenForm "2-0-1-0-fm-QueryMenu"
```

End Sub

```
=====
Function/Sub Name: lblQuery_MouseMove()
Description: Sets command button text colors.
Input: None
```

```

'
'Output: None
'
'References: None
'
=====
Private Sub lblQuery_MouseMove(Button As Integer, Shift
As Integer, X As Single, Y As Single)

    ' Change menu color when mouse is over button
    Me.lblQuery.BackColor = QBColor(9) ' Blue
    Me.lblGraph.BackColor = QBColor(15) ' White
    Me.lblReport.BackColor = QBColor(15) ' White
    Me.lblAddEditMishaps.BackColor = QBColor(15) ' White
    Me.lblInvestigate.BackColor = QBColor(15) ' White
    Me.lblExit.BackColor = QBColor(15) ' White
    '

    Me.lblQuery.SpecialEffect = 1 ' Raised
    Me.lblGraph.SpecialEffect = 0 ' Normal
    Me.lblReport.SpecialEffect = 0 ' Normal
    Me.lblAddEditMishaps.SpecialEffect = 0 ' Normal
    Me.lblInvestigate.SpecialEffect = 0 ' Normal
    Me.lblExit.SpecialEffect = 0 ' Normal
    '

    Me.lblQuery.ForeColor = QBColor(15) ' White
    Me.lblGraph.ForeColor = QBColor(0) ' Black
    Me.lblReport.ForeColor = QBColor(0) ' Black
    Me.lblAddEditMishaps.ForeColor = QBColor(0) ' Black
    Me.lblInvestigate.ForeColor = QBColor(0) ' Black
    Me.lblExit.ForeColor = QBColor(0) ' Black

End Sub

'
=====
'Function/Sub Name: lblReport_Click()
'
'Description: Opens the Report form (8-0-0-1-fm-Reports).
'
'Input: None
'
'Output: None
'
'References:
'      - 8-0-0-1-fm-Reports
'
=====
Private Sub lblReport_Click()

    DoCmd.OpenForm "8-0-0-1-fm-Reports"

End Sub

'
=====
'Function/Sub Name: lblReport_MouseMove()
'
'Description: Sets command button text colors.
'
'Input: None
'
'Output: None
'
'References: None

```

```

'
=====
Private Sub lblReport_MouseMove(Button As Integer, Shift
As Integer, X As Single, Y As Single)

    ' Change menu color when mouse is over button
    Me.lblQuery.BackColor = QBColor(15) ' White
    Me.lblGraph.BackColor = QBColor(15) ' White
    Me.lblReport.BackCobr = QBColor(9) ' Blue
    Me.lblAddEditMishaps.BackColor = QBColor(15) ' White
    Me.lblInvestigate.BackColor = QBColor(15) ' White
    Me.lblExit.BackColor = QBColor(15) ' White
    '

    Me.lblQuery.SpecialEffect = 0 ' Normal
    Me.lblGraph.SpecialEffect = 0 ' Normal
    Me.lblReport.SpecialEffect = 1 ' Raised
    Me.lblAddEditMishaps.SpecialEffect = 0 ' Normal
    Me.lblInvestigate.SpecialEffect = 0 ' Normal
    Me.lblExit.SpecialEffect = 0 ' Normal
    '

    Me.lblQuery.ForeColor = QBColor(0) ' Black
    Me.lblGraph.ForeColor = QBColor(0) ' Black
    Me.lblReport.ForeColor = QBColor(15) ' White
    Me.lblAddEditMishaps.ForeColor = QBColor(0) ' Black
    Me.lblInvestigate.ForeColor = QBColor(0) ' Black
    Me.lblExit.ForeColor = QBColor(0) ' Black

End Sub

'
=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'      - clFormWindow
'
=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        .Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

End Sub

```

## **FORMCLASS-PleaseWait**

Option Explicit

'#####

' FORM DESCRIPTION

'#####

'Class Name: PleaseWait

'

'Author: Pat Flanders & Scott Tufts

'

'Description: This class is the splash screen that user sees at  
'program initiation. It is responsible for setting global  
properties  
'for the session at startup.

'

'References: None

'

'#####

'\*\*\*\*\*

' FUNCTIONS

'\*\*\*\*\*

'=====

'Function/Sub Name: Command17\_Click()

'

'Description: Closes the form. This button is not visible  
during  
'normal program operation and must be turned on in design  
view  
'to use it. It is provided for troubleshooting connection  
problems  
'which often result in a "hang" at this screen with now way to  
'terminate program execution unless this button is enabled.

'

'Input: None

'

'Output: None

'

'References: None

'

'=====

Private Sub Command17\_Click()

DoCmd.Close acForm, "PleaseWait"

End Sub

'=====

'Function/Sub Name: Form\_Load()

'

'Description: Sets the global properties for the session. This  
includes

'application icon, margins, and other default behaviors.

'Input: None

'

'Output: None

'

'References: None

'

'=====

Private Sub Form\_Load()

Screen.MousePointer = 11

'Determine OS and store value in a global variable

'A value of 2 or higher means WIN 2K or WIN NT

Dim myVer As OSVERSIONINFO

Dim q As Long

myVer.dwOSVersionInfoSize = 148

q& = GetVersionEx(myVer)

'Uncomment this line for complet o/s version description  
information

'MsgBox "Platform ID = " & myVer.dwPlatformId & ",

Version = " & myVer.dwMajorVersion & "." &

myVer.dwMinorVersion & " Build " &

(myVer.dwBuildNumber And &HFFFF&)

GlobalDeclarations.gStrOSType = myVer.dwPlatformId

'Set the application icon

CurrentProject.Properties.Add "AppIcon",

Application.CurrentProject.Path & "\hfacs.ico"

CurrentProject.Properties("AppIcon") =

Application.CurrentProject.Path & "\hfacs.ico"

Application.RefreshTitleBar

DoEvents 'Redraw screen

'Set program GLOBAL start-up options.

Application.SetOption "Show Startup Dialog Box", False

Application.SetOption "Left Margin", 1

Application.SetOption "Right Margin", 1

Application.SetOption "Top Margin", 1

Application.SetOption "Bottom Margin", 1

Application.SetOption "Default Find/Replace Behavior", 1

Application.SetOption "Behavior Entering Field", 1

Application.SetOption "Show Windows In TaskBar", False

DoEvents 'Redraw screen

Screen.MousePointer = 0

End Sub

## **MODULE-ConenctionFunctions**

```
Option Compare Database
Option Explicit
'#####
'      MODULE DESCRIPTION
'#####
'Class Name: ConnectionFunctions.bas
'
'Author: Pat Flanders & Scott Tufts
'
'Description: This module contains the vast majority of the
'helper
'functions used by the program. It contains functions for
'connecting
'and disconnecting the application to a SQL server, replacing
'the
'database via FTP and disk file, toggling database type,
'printing
'the MS Chart graphs from the windows clipboard, as well as,
'all command bar functions and command bar menu scripts.
'
'References:
'      - HFACS.dll
'      - HFACSClipboard.dll
'#####
```

```
*****
'
'      FUNCTIONS
'
*****
```

```
'=====
'Function/Sub Name: CreateConnection()
'
'Description: Connects the application to a SQL server and
'provides
'the interface for the HFACS.dll. Reads the initial values for
'most global program variables from the HFACS.ini file via
'the
'HFACS.dll and the SQL Server that becomes connected.
'Verifies the
'database type and ensure that the Server being connected to
'is of
'the proper type (military vice civilian).
'
'Input: None
'
'Output: None
'
'References:
'      - HFACS.dll
'
'=====
```

```
Public Sub CreateConnection()
```

```
    On Error GoTo startError
```

```
    Set oHFACSConnection = New HFACSConnection
    Dim bConnResults As Boolean
    bConnResults = False
```

```
    'Read in the values from the .dll
```

```
    gStrUID = oHFACSConnection.User
    gStrPWD = oHFACSConnection.Password
    gStrServerName = oHFACSConnection.ServerName
    gStrDatabaseFileName =
oHFACSConnection.DatabaseFileName
    gStrDatabaseName = oHFACSConnection.DatabaseName
    gStrAppPath = oHFACSConnection.AppPath
    gStrAutoLogon = oHFACSConnectionAutomaticLogon
    gStrFirstRun = oHFACSConnection.FirstRunCheck
    gStrNTauth = oHFACSConnection.UseNTAuth
    gStrTypeDB = oHFACSConnection.TypeDatabase
    gTheConnectionString =
oHFACSConnection.ConnectionString
```

```
StartWrongTypConnMade:
```

```
    While bConnResults = False
```

```
        bConnResults = oHFACSConnection.doConnect
```

```
StartLogon:
```

```
    If bConnResults = False Then
        Dim response As Variant
        DoCmd.Beep
        response = MsgBox("An error occurred while
trying to connect to the server." & Chr(13) & Chr(13) &
"You must connect to a server in order to use the HFACS
database." & Chr(13) & Chr(13) & "You can RETRY by
specifying new logon information or CANCEL and exit to
Windows.", vbRetryCancel + vbExclamation +
vbDefaultButton1, "Problem With Connection")
```

```
    If response = vbCancel Then
        'Exit to Windows
        Set oHFACSConnection = Nothing
        ConnectionFunctions.removeConnection
```

```
    Else
        'Logon with prompt
        bConnResults =
oHFACSConnection.doConnect(PROMPT)
    End If
    End If
Wend
```

```
    'Reset all the local global variables to capture changes
made during the
'logon process.
    gStrUID = oHFACSConnection.User
    gStrPWD = oHFACSConnection.Password
    gStrServerName = oHFACSConnection.ServerName
    gStrDatabaseFileName =
oHFACSConnection.DatabaseFileName
    gStrDatabaseName = oHFACSConnection.DatabaseName
    gStrAppPath = oHFACSConnection.AppPath
    gStrAutoLogon = oHFACSConnectionAutomaticLogon
    gStrFirstRun = oHFACSConnection.FirstRunCheck
    gStrNTauth = oHFACSConnection.UseNTAuth
    gStrTypeDB = oHFACSConnection.TypeDatabase
    gTheConnectionString =
oHFACSConnection.ConnectionString
```

```
    Application.CurrentProject.OpenConnection
    GlobalDeclarations.gTheConnectionString
```



```

If Application.CurrentProject.IsConnected = False Then
GoTo StartLogon

'Run stored procedure to make sure you are connecting to
the right type
'database (military or civilian).
'Declare objects for querying a stored procedure to get the
new record
Dim cnn As Connection
Dim oCmd As ADODB.Command
Dim rst As ADODB.Recordset

Set cnn = CurrentProject.Connection
cnn.CursorLocation = adUseClient
Set rst = New ADODB.Recordset
Set oCmd = New ADODB.Command
oCmd.ActiveConnection = cnn
oCmd.CommandText = """"9-0-0-1-flanLookupDBType""""
oCmd.CommandType = adCmdStoredProc

'Run the SP
Set rst = oCmd.Execute

'Get the record count
rst.MoveFirst
Dim tempString As String
tempString = rst!DatabaseType 'Get the database type

'Clean up
rst.Close
Set oCmd = Nothing
cnn.Close

'MsgBox "Global: " & GlobalDeclarations.gStrTypeDB &
" Read From Remote DB: " & Trim(tempString)
If GlobalDeclarations.gStrTypeDB <> Trim(tempString)
Then
    Dim sTempType As String
    If Trim(tempString) = "C" Then
        sTempType = "CIVILIAN but this version of HFACS
is configured for MILITARY. "
    Else
        sTempType = "MILITARY but this version of
HFACS is configured for CIVILIAN. "
    End If
    MsgBox "You are trying to connect to a database
configured for " & sTempType & _
Chr(13) & Chr(13) & "Please connect to another
server.", vbOKOnly + vbExclamation, _
"Can't Connect To That Type Database"
    bConnResults = False
    GoTo StartWrongTypConnMade
End If

exitSub:
Exit Sub

startError:
MsgBox Err.Description
MsgBox Err.Number
bConnResults = False
Resume StartLogon

End Sub
'=====

```

```

Function/Sub Name: InitConnection()
'
'Description: Disables the Access "close" button on the main
access
>window, preventing users from improperly shutting down the
the
'application. Launches the "PleaseWait" form while the
connection
'to the SQL server is initialized, giving the illusion of
'separate threads of execution and providing the user a screen
'to look at during this long process.
'
'Input: None
'
'Output: None
'
'References:
'         - PleaseWait Form
'         - CloseCommand Class
'=====

Function InitConnection()
    On Error GoTo startError

    'Disable the Access master window clos control button
    Dim c As CloseCommand
    Set c = New CloseCommand
    'Disable Close menu.
    c.Enabled = False

    DoCmd.OpenForm "PleaseWait", acNormal, "", "",
acReadOnly, acNormal
    DoCmd.RepaintObject acForm, "PleaseWait"
    ConnectionFunctions.CreateConnection
    DoCmd.Close acForm, "PleaseWait"

exitSub:
Exit Function

startError:
Resume exitSub

End Function

'=====
Function/Sub Name: changeServer()
'
'Description: Provides the functionality to change server
connections
'via the HFACS.dll.
'
'Input: None
'
'Output: Success or failure.
'
'References:
'         - HFACS.dll
'=====

Public Function changeServer() As Boolean

    Dim bResult As Boolean

StartWrongTypConnMade:

    'Bring up the logon prompt
    bResult = oHFACSConnection.doConnect(PROMPT)

```

```

If bResult = True Then

    'Reset all the local global variables to capture changes
    made during the
    'login process.
    gStrUID = oHFACSCConnection.User
    gStrPWD = oHFACSCConnection.Password
    gStrServerName = oHFACSCConnection.ServerName
    gStrDatabaseFileName =
oHFACSCConnection.DatabaseFileName
    gStrDatabaseName =
oHFACSCConnection.DatabaseName
    gStrAppPath = oHFACSCConnection.AppPath
    gStrAutoLogin =
oHFACSCConnectionAutomaticLogin
    gStrFirstRun = oHFACSCConnection.FirstRunCheck
    gStrNTauth = oHFACSCConnection.UseNTAuth
    gStrTypeDB = oHFACSCConnection.TypeDatabase
    gTheConnectionString =
oHFACSCConnection.ConnectionString

    Application.CurrentProject.OpenConnection
    GlobalDeclarations.gTheConnectionString

    'Run stored procedure to make sure you are connecting
    to the right type
    'database (military or civilian).
    'Declare objects for querying a stored procedure to get
    the new record
    Dim cnn As Connection
    Dim oCmd As ADODB.Command
    Dim rst As ADODB.Recordset

    Set cnn = CurrentProject.Connection
    cnn.CursorLocation = adUseClient
    Set rst = New ADODB.Recordset
    Set oCmd = New ADODB.Command
    oCmd.ActiveConnection = cnn
    oCmd.CommandText = """"9-0-0-1-
flanLookupDBType""""
    oCmd.CommandType = adCmdStoredProc

    'Run the SP
    Set rst = oCmd.Execute

    'Get the record count
    rst.MoveFirst
    Dim tempString As String
    tempString = rst!DatabaseType 'Get the database type

    'Clean up
    rst.Close
    Set oCmd = Nothing
    cnn.Close

    'MsgBox "Global: " & GlobalDeclarations.gStrTypeDB
    & " Read From Remote DB: " & Trim(tempString)
    If GlobalDeclarations.gStrTypeDB <>
Trim(tempString) Then
        Dim sTempType As String
        If Trim(tempString) = "C" Then
            sTempType = "CIVILIAN but this version of
HFACS is configured for MILITARY. "
        Else
            sTempType = "MILITARY but this version of
HFACS is configured for CIVILIAN. "

```

```

End If
MsgBox "You are trying to connect to a database
configured for " & sTempType & _
Chr(13) & Chr(13) & "Please connect to another
server.", vbOKOnly + vbExclamation, _
"Can't Connect To That Type Database"
bResult = False
GoTo StartWrongTypConnMade
End If

Form_MainMenu.lblServerConnectedTo.Caption =
"Connected To Server: " &
GlobalDeclarations.gStrServerName
Form_MainMenu.Refresh

changeServer = True
Else
changeServer = False
End If

End Function

```

```

'=====
'Function/Sub Name:  getUpdateFTP()
'
'Description: Provides the functionality replace the database
on the
'local SQL server via an FTP process. THE USER MUST
BE LOGGED ON
'WITH THE SA ACCOUNT, BEING AN
ADMINISTRATOR IS NOT ENOUGH.
'
'Input: None
'
'Output: Success or failure.
'
'References:
'      - HFACS.dll
'=====
Public Function getUpdateFTP() As Boolean

    If GlobalDeclarations.gStrUID <> "sa" Then
        MsgBox "You must be logged on as SA to replace the
database", vbOKOnly + vbExclamation, "User Is Not SA"
        getUpdateFTP = False
        Exit Function
    End If

    Dim i As Integer

    getUpdateFTP = oHFACSCConnection.getUpdateFTP

    On Error GoTo startError
    If getUpdateFTP = True Then
        Application.CurrentProject.CloseConnection 'Close the
Connection
        Application.CurrentProject.OpenConnection 'Set the
connection to nothing
        DoCmd.OpenForm "MainMenu"
        getUpdateFTP = True
    Else
        getUpdateFTP = False
        ConnectionFunctions.CreateConnection
    End If

exitSub:

```

```

' Set oHFACSCConnection = Nothing
Exit Function

startError:
' This block of code is required to get the connection to
close.
' It is a documented MS Access 2000 bug.
i = i + 1
If i < 99 Then 'Continue trying to close connection.
    DoEvents
    Resume
End If

Resume exitSub

End Function

'=====
'Function/Sub Name: getUpdateFromDisk()
'
'Description: Provides the functionality replace the database
on the
'local SQL server via an file on a CD or network share
process.
'THE USER MUST BE LOGGED ON WITH THE SA
ACCOUNT, BEING AN ADMINISTRATOR
'IS NOT ENOUGH.
'
'Input: None
'
'Output: Success or failure.
'
'References:
'      - HFACS.dll
'
'=====
Public Function getUpdateFromDisk() As Boolean

    If GlobalDeclarations.gStrUID <> "sa" Then
        MsgBox "You must be logged on as SA to replace the
database", vbOKOnly + vbExclamation, "User Is Not SA"
        getUpdateFromDisk = False
        Exit Function
    End If

    Dim i As Integer 'Counter

    getUpdateFromDisk =
oHFACSCConnection.getUpdateDisk

    On Error GoTo startError
    If getUpdateFromDisk = True Then
        Application.CurrentProject.CloseConnection 'Close the
Connection
        Application.CurrentProject.OpenConnection 'Set the
connection to nothing
        DoCmd.OpenForm "MainMenu"
        getUpdateFromDisk = True
    Else
        getUpdateFromDisk = False
        ConnectionFunctions.CreateConnection
    End If

exitSub:
    Exit Function

startError:

```

```

' This block of code is required to get the connection to
close.
' It is a documented MS Access 2000 bug.
i = i + 1
If i < 99 Then 'Continue trying to close connection.
    DoEvents
    Resume
End If

Resume exitSub

End Function

'=====
'Function/Sub Name: removeConnection()
'
'Description: Properly disconnects the application from the
SQL
'server and terminates the Access session.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Public Function removeConnection()

    Dim i As Integer 'Counter
    On Error GoTo startError
    Application.CurrentProject.CloseConnection 'Close the
Connection
    Application.CurrentProject.OpenConnection 'Set the
connection n to nothing
    Set oHFACSCConnection = Nothing

exitSub:
    Application.CommandBars("mnuProgramMain").Visible =
False
    DoCmd.Quit
    removeConnection = True
    Exit Function

startError:
' This block of code is required to get the connection to
close.
' It is a documented MS Access 2000 bug.
i = i + 1
If i < 99 Then 'Continue trying to close connection.
    DoEvents
    Resume
End If

Resume exitSub

End Function

'=====
'Function/Sub Name: CommandbarEnable()
'
'Description: Allows manipulation of command (menu bars).
'
'This function has four arguments:
'

```

```

'Cmdbar is a CommandBar object that represents the
command
'bar containing the menu item to be enabled or disabled.
'
'CmdBarEnabled is a Boolean value in which you pass
"True"
'or "False" in order to enable or disable the menu item being
'manipulated.
'
'TopLevel is an integer representing the index of the Top-
level
'menu item being manipulated.
'
'Sublevel is an optional integer representing the index of the
'menu item being manipulated under the Top-level menu
item.
'
'Example: To disable only the "File" menu item on the
'"NorthwindCustomMenuBar" command bar, use the
following:
'
'CommandbarEnable(Commandbars("NorthwindCustomMen
uBar"),False,1)
'
'Example2: To disable the "Get external Data" Menu item
under
'the "File" menu item on the "NorthwindCustomMenuBar"
command
'bar, use the following:
'
'CommandbarEnable(Commandbars("NorthwindCustomMen
uBar"),False,1,3)
'
'To "re-enable" the same menu item, use the following:
'
'CommandbarEnable(Commandbars("NorthwindCustomMen
uBar"),True,1,3)
'
'=====
Public Function CommandbarEnable(Cmdbar As
CommandBar, _
    CmdbarEnabled As Boolean, TopLevel As Integer, _
    Optional Sublevel As Integer)

    Dim SubCommandbar

    On Error GoTo Err_CommandBarEnable

    'If the command bar is not visible, make it so.
    If Cmdbar.Visible = False Then Cmdbar.Visible = True

    'If no menu item on a submenu is selected for
    enabling/disabling,
    'enable/disable the top level menu choice only.
    If IsMissing(Sublevel) Or Sublevel = 0 Then
        Cmdbar.Controls(TopLevel).Enabled =
Cmdbar.Enabled
        'If a menu item on a submenu is selected for
        'enabling/disabling, do so now.
    Else
        Set SubCommandbar =
Cmdbar.Controls(TopLevel)
        SubCommandbar.Controls(Sublevel).Enabled =
Cmdbar.Enabled
    End If

Exit_CommandBarEnable:
    Exit Function

```

```

Err_CommandBarEnable:
    MsgBox "Error " & CStr(ERR) & " " &
ERR.Description & _
    " has occurred in the CommandBarEnable Function",
vbOKOnly, _
    "Error Detected"
    Resume Exit_CommandBarEnable

End Function

'=====
Function/Sub Name: toggleDBType()
'
'Description: Properly disconnects the application from the
SQL
'server and terminates the Access session.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Public Function toggleDBType() As Boolean

    Dim response As Variant
    Dim sDBType As String

    If GlobalDeclarations.gStrTypeDB = "C" Then
        sDBType = "Civilian to Military. "
    Else
        sDBType = "Military to Civilian. "
    End If

    DoCmd.Beep
    response = MsgBox("You are about to toggle this database
from " & sDBType & Chr(13) & Chr(13) & "This may
require you to reconnect to HFACS." & Chr(13) & Chr(13)
& "Do you wish to continue?", vbYesNo + vbQuestion +
vbDefaultButton2, "Toggle Database Type?")

    If response = vbYes Then

        'Declare objects for querying a stored procedure to get
the new record
        Dim rsTheNewMishap As New Recordset
        Dim commandADO As New ADODB.Command
        Dim conADO As New ADODB.Connection

        ' This is where we create the Connection object.
        Set conADO = CurrentProject.Connection

        If GlobalDeclarations.gStrTypeDB = "C" Then
            GlobalDeclarations.gStrTypeDB = "M"
            rsTheNewMishap.Open "UPDATE tblDatabaseType
SET tblDatabaseType.DatabaseType = 'M' WHERE
tblDatabaseType.DatabaseType = 'C'", conADO,
adOpenDynamic, adLockOptimistic, adCmdText
            oHFACSConnection.TypeDatabase = "M"
        Else
            GlobalDeclarations.gStrTypeDB = "C"
            rsTheNewMishap.Open "UPDATE tblDatabaseType
SET tblDatabaseType.DatabaseType = 'C' WHERE
tblDatabaseType.DatabaseType = 'M'", conADO,
adOpenDynamic, adLockOptimistic, adCmdText

```

```

        oHFACSCConnection.TypeDatabase = "C"
    End If

    oHFACSCConnection.writeINIFile

    'Destroy objects used for the query
    Set commandADO = Nothing
    Set conADO = Nothing
    Set rsTheNewMishap = Nothing

    Dim i As Integer ' counter

    On Error GoTo startError
    Application.CurrentProject.CloseConnection 'Close the
Connection
    Application.CurrentProject.OpenConnection 'Set the
connection to nothing
    DoCmd.OpenForm "MainMenu"

    toggleDBType = True
    GoTo exitSub

End If

toggleDBType = False

exitSub:
    Exit Function

startError:
    'This block of code is required to get the connection to
close.
    'It is a documented MS Access 2000 bug.
    i = i + 1
    If i < 99 Then 'Continue trying to close connection.
        DoEvents
        Resume
    End If

    Resume exitSub

End Function

'=====
Function/Sub Name: copyGraphToClipboard()
'
'Description: Copies the MS Chart object on form 4-0-1-2-
frm-TheActualGraph
'to the windows clipboard.
'
'Input: None
'
'Output: Success or failure.
'
'References:
'
'    - 4-0-1-2-firm-TheActualGraph
'
'=====
Public Function copyGraphToClipboard() As Boolean

    ' Call the EditCopy method to send the chart to the
clipboard
    Forms![4-0-1-2-firm-
TheActualGraph].chtTheGraph.EditCopy

    'For future use. At this point you could
'save the data on the clipboard as a bitmap

```

```

'you can also use clipboard viewer to see everything
'SavePicture Clipboard.GetData, "c:\test1.bmp"

copyGraphToClipboard = True

End Function

'=====
Function/Sub Name: toggleXLabels()
'
'Description: Toggles the X axis values visible/hidden for the
'MS Chart object on form 4-0-1-2-firm-TheActualGraph.
'
'Input: None
'
'Output: Success or failure.
'
'References:
'
'    - 4-0-1-2-firm-TheActualGraph
'
'=====
Public Function toggleXLabels() As Boolean

    'Toggle visibility of X-Axis labels
    If Forms![4-0-1-2-firm-
TheActualGraph].chtTheGraph.Plot.Axis(VtChAxisIdX).Axi
sScale.Hide = False Then
        Forms![4-0-1-2-firm-
TheActualGraph].chtTheGraph.Plot.Axis(VtChAxisIdX).Axi
sScale.Hide = True
    Else
        Forms![4-0-1-2-firm-
TheActualGraph].chtTheGraph.Plot.Axis(VtChAxisIdX).Axi
sScale.Hide = False
    End If

    toggleXLabels = True

End Function

'=====
Function/Sub Name: toggleYLabels()
'
'Description: Toggles the Y axis values visible/hidden for the
'MS Chart object on form 4-0-1-2-firm-TheActualGraph.
'
'Input: None
'
'Output: Success or failure.
'
'References:
'
'    - 4-0-1-2-firm-TheActualGraph
'
'=====
Public Function toggleYLabels() As Boolean

    'Toggle visibility of Y-Axis labels
    If Forms![4-0-1-2-firm-
TheActualGraph].chtTheGraph.Plot.Axis(VtChAxisIdY).Axi
sScale.Hide = True Then
        Forms![4-0-1-2-firm-
TheActualGraph].chtTheGraph.Plot.Axis(VtChAxisIdY).Axi
sScale.Hide = False
    Else
        Forms![4-0-1-2-firm-
TheActualGraph].chtTheGraph.Plot.Axis(VtChAxisIdY2).A
xisScale.Hide = False
    End If

```

```

Else
    Forms![4-0-1-2-firm-
TheActualGraph].chtTheGraph.Plot.Axis(VtChAxisIdY).AxisScale.Hide = True
    Forms![4-0-1-2-firm-
TheActualGraph].chtTheGraph.Plot.Axis(VtChAxisIdY2).AxisScale.Hide = True
End If

toggleYLabels = True

End Function

```

```

=====
'Function/Sub Name: sendClipToPrinter()
'
'Description: Prints the MS Chart object on form 4-0-1-2-firm-TheActualGraph.
'
'Input: None
'
'Output: Success or failure.
'
'References:
'    - 4-0-1-2-firm-TheActualGraph
'    - HFACSClipboard.dll
'
=====
Public Function sendClipToPrinter() As Boolean

    On Error GoTo startError

    'Copy the graph to the clipboard
    DoCmd.RunMacro "macroMnuCopyGraphToClipboard"

    'Print small graphs portrait and large ones landscape
    Dim oMyClipObject As New clsClipBoard
    If Forms![4-0-1-2-firm-TheActualGraph].togEnlarge.Value = -1 Then
        oMyClipObject.clipOutLandscape
    Else
        oMyClipObject.clipOutPortrait
    End If

    sendClipToPrinter = True

exitSub:
    Set oMyClipObject = Nothing

```

Exit Function

```

startError:
    MsgBox "There was a problem with your default printer. Check to ensure that it is on-line and loaded with paper and try printing again.", vbOKOnly + vbExclamation, "Problem Printing"
    sendClipToPrinter = False
    Resume exitSub

```

End Function

```

=====
'Function/Sub Name: waitScreen()
'
'Description: Shows the please wait screen with spinning globe while calculating report data.
'
'Input: None
'
'Output: None
'
'References:
'    - 7-0-0-1-PopUpFrm-waitProgressBar
'
=====
Function waitScreen(sReportName As String) As Boolean
    On Error GoTo startError

    DoCmd.OpenForm "7-0-0-1-PopUpFrm-waitProgressBar", acNormal, "", "", acReadOnly, acNormal
    DoCmd.RepaintObject acForm, "7-0-0-1-PopUpFrm-waitProgressBar"
    DoCmd.OpenReport sReportName, acViewPreview
    DoCmd.Close acForm, "7-0-0-1-PopUpFrm-waitProgressBar"

    waitScreen = True

exitSub:
    Exit Function

startError:
    waitScreen = False
    Resume exitSub

End Function

```

## **MODULE-DeterminesOSDeclares**

Option Explicit

```
Type OSVERSIONINFO
dwOSVersionInfoSize As Long
dwMajorVersion As Long
dwMinorVersion As Long
dwBuildNumber As Long
dwPlatformId As Long
szCSDVersion As String * 128 ' Maintenance string for PSS
usage
End Type
Declare Function GetVersionEx Lib "kernel32" Alias
"GetVersionExA" (lpVersionInformation As
OSVERSIONINFO) As Long
Declare Function GetSystemMetrics Lib "user32" (ByVal
nIndex As Long) As Long
Public Const SM_CLEANBOOT = 67
Public Const SM_DEBUG = 22
Public Const SM_SLOWMACHINE = 73
Public Const VER_PLATFORM_WIN32s = 0
Public Const VER_PLATFORM_WIN32_WINDOWS = 1
Public Const VER_PLATFORM_WIN32_NT = 2
```

```
#####
'          MODULE DESCRIPTION
#####
'Class Name: DetermineOSDeclares.bas
'
```

'Author: Pat Flanders & Scott Tufts

'Description: Contains various functions for determining system properties like O/S type and version of Access that is running.

'The O/S type functions are declared above and result in direct querying of the Windows API.

'References: None

```
#####
```

```
*****
'          FUNCTIONS
*****
```

```
=====
Function/Sub Name: IsRuntime()
'
```

'Description: Determines if Access runtime is being used to run the application. Access runtime has no support for reports.

'Input: None

'Output: Success or failure.

'References: None

```
=====
Function IsRuntime() As Boolean
```

' Check if this application is using the run-time version of Access.  
IsRuntime = SysCmd(acSysCmdRuntime)

End Function

```
=====
Function/Sub Name: IsRunning()
'
```

'Description: To prevent a second instance from loading if a user mistakenly attempts to launch it twice. This code is called from the autoexec macro to test whether the app is already running and terminate the launch if a copy of it is already open.

'Input: None

'Output: -1 means that an instance is already running.

'References: None

```
=====
Function IsRunning() As Integer
```

If TestDDELink(Application.CurrentProject.Name) Then  
'A -1 means that this is a second instance.  
IsRunning = -1

Else  
IsRunning = 0

End If

End Function

'Helper Function for IsRunning() above

```
Function TestDDELink(ByVal strAppName$) As Integer
```

```
Dim varDDEChannel As Variant
On Error Resume Next
Application.SetOption ("Ignore DDE Requests"), True
varDDEChannel = DDEInitiate("MSAccess",
strAppName)
```

'When the app isn't already running this will error

If ERR Then

TestDDELink = False

Else

TestDDELink = True

DDETerminate varDDEChannel

DDETerminateAll

End If

Application.SetOption ("Ignore DDE Requests"), False

End Function

## **MODULE-ezSizingFunctions**

Option Compare Database

Option Explicit

'#####  
'          MODULE DESCRIPTION

'#####

'Class Name: ezSizingFunctions.bas

'

'Author: EZ Sizing Functions

'    Copyright (C) 2000 Database Creations, Inc.

'    Revision 6/14/00

'    based on 8/25/99 code with revisions

'Description: Contains various functions for dynamically  
resizing

'the forms in the application based on the user's screen  
resolution.

'

'

'References: None

'

'#####

'\*\*\*\*\*

'          FUNCTIONS

'\*\*\*\*\*

'Functions are defined below by the author and are Copyright  
of

'Database Creations, Inc.

Type RECT

    x1 As Long

    y1 As Long

    x2 As Long

    y2 As Long

End Type

Type TEXTMETRIC

    tmHeight As Integer

    tmAscent As Integer

    tmDescent As Integer

    tmInternalLeading As Integer

    tmExternalLeading As Integer

    tmAveCharWidth As Integer

    tmMaxCharWidth As Integer

    tmWeight As Integer

    tmItalic As String \* 1

    tmUnderlined As String \* 1

    tmStruckOut As String \* 1

    tmFirstChar As String \* 1

    tmLastChar As String \* 1

    tmDefaultChar As String \* 1

    tmBreakChar As String \* 1

    tmPitchAndFamily As String \* 1

    tmCharSet As String \* 1

    tmOverhang As Integer

    tmDigitizedAspectX As Integer

    tmDigitizedAspectY As Integer

End Type

Declare Function IsZoomed Lib "user32" (ByVal hwnd As  
Long) As Long

Declare Function IsIconic Lib "user32" (ByVal hwnd As  
Long) As Long

Declare Function GetDesktopWindow Lib "user32" () As  
Long

Declare Function GetWindowRect Lib "user32" (ByVal  
hwnd As Long, rectangle As RECT) As Long

Declare Function GetTextMetrics Lib "gdi32" Alias

"GetTextMetricsA" (ByVal hdc As Long, lpMetrics As

TEXTMETRIC) As Long

Declare Function GetWindowDC Lib "user32" (ByVal hwnd  
As Long) As Long

Declare Function ReleaseDC Lib "user32" (ByVal hwnd As  
Long, ByVal hdc As Long) As Long

Declare Function SetMapMode Lib "gdi32" (ByVal hdc As  
Long, ByVal nMapMode As Long) As Long

Public Sub ezSizeForm(xForm As Form, ScaleFactor As  
Single, Optional EchoOff As Boolean = True)

'This subroutine will resize the form specified by parameter  
xForm by the factor of ScaleFactor

'If scale factor is 0 or negative, automatic scaling will occur  
based on the following

'  Value  Forms originally designed for

'  0      640 x 480

'  -1     800 x 600

'  -2     1024 x 768

'  -3     1280 x 1024

'  -4     1600 x 1200

'  -5     1152 x 864 OR 1152 x 870

Dim ActiveForm As Object

Dim i As Integer

Dim D(200, 3) As Single

    On Error GoTo errorHandler

    If ScaleFactor = 1 Then GoTo Done

    If ScaleFactor <= 0 Then ScaleFactor =

ezGetScaleFactor(ScaleFactor)

    If EchoOff Then DoCmd.Echo False

    Set ActiveForm = xForm

    'If form in datasheet view then don't resize

    If xForm.CurrentView <> 1 Then GoTo Done

    'If the form is maximized then don't resize

    If IsZoomed(xForm.hwnd) <> 0 Then GoTo Done

    With ActiveForm

        If ScaleFactor > 1 Then 'form is growing

        'deal with section heights and form width first

        On Error Resume Next 'handle error for non-existent  
sections

        For i = 0 To 4

            .Section(i).Height = .Section(i).Height \*

ScaleFactor

        Next i

        On Error GoTo errorHandler

        .Width = .Width \* ScaleFactor

    End If

    'save old dimensions of subforms/groups/tabs

    For i = 0 To .Count - 1

        Select Case .Controls(i).ControlType



```

        Case acOptionGroup, acSubform, acTabCtl
            D(i, 0) = .Controls(i).Width
            D(i, 1) = .Controls(i).Height
            D(i, 2) = .Controls(i).Left
            D(i, 3) = .Controls(i).Top
        End Select
    Next i

    'deal with controls
    For i = 0 To .Count - 1
        Select Case .Controls(i).ControlType
            Case acOptionGroup, acPage
                'do nothing now
            Case acTabCtl
                .Controls(i).TabFixedWidth =
                .Controls(i).TabFixedWidth * ScaleFactor
                .Controls(i).TabFixedHeight =
                .Controls(i).TabFixedHeight * ScaleFactor
                If .Controls(i).Left < 0 Then .Controls(i).Left = 0
                .Controls(i).Left = .Controls(i).Left * ScaleFactor
                .Controls(i).Top = .Controls(i).Top * ScaleFactor
                .Controls(i).Width = .Controls(i).Width *
                ScaleFactor
                .Controls(i).Height = .Controls(i).Height *
                ScaleFactor
                .Controls(i).fontsize = .Controls(i).fontsize *
                ScaleFactor
            Case acSubform
                On Error Resume Next
                ezSizeForm .Controls(i).Form, ScaleFactor,
                False
                On Error GoTo errorHandler
            Case Else
                On Error Resume Next
                If .Controls(i).Left < 0 Then .Controls(i).Left = 0
                .Controls(i).Left = .Controls(i).Left *
                ScaleFactor
                .Controls(i).Top = .Controls(i).Top *
                ScaleFactor
                .Controls(i).Width = .Controls(i).Width *
                ScaleFactor
                .Controls(i).Height = .Controls(i).Height *
                ScaleFactor
                .Controls(i).fontsize = .Controls(i).fontsize *
                ScaleFactor
                On Error GoTo errorHandler
        End Select
    Next i

    'fix dimensions of subforms/groups/tabs
    If ScaleFactor > 1 Then
        On Error Resume Next
        For i = 0 To 4
            .Section(i).Height = .Section(i).Height * ScaleFactor
        Next i
        On Error GoTo errorHandler
    End If
    For i = 0 To .Count - 1
        Select Case .Controls(i).ControlType
            Case acSubform
                .Controls(i).Width = D(i, 0) * ScaleFactor
                .Controls(i).Height = D(i, 1) * ScaleFactor
                .Controls(i).Left = D(i, 2) * ScaleFactor
                .Controls(i).Top = D(i, 3) * ScaleFactor
        End Select
    Next i
    For i = 0 To .Count - 1
        Select Case .Controls(i).ControlType

```

```

        Case acOptionGroup, acTabCtl
            .Controls(i).Left = D(i, 2) * ScaleFactor
            .Controls(i).Top = D(i, 3) * ScaleFactor
            .Controls(i).Width = D(i, 0) * ScaleFactor
            .Controls(i).Height = D(i, 1) * ScaleFactor
        End Select
    Next i

    'Resize form dimensions and fit window to form
    On Error Resume Next
    For i = 0 To 4
        .Section(i).Height = 0
    Next i
    On Error GoTo errorHandler
    .Width = 0
    DoCmd.RunCommand acCmdSizeToFitForm
    GoTo Done

errorHandler:
    If ERR.Number = 2046 Then GoTo Done
    MsgBox "Error with control " & .Controls(i).Name &
    vbCrLf & _
    "L: " & .Controls(i).Left & " - " & D(i, 2) & vbCrLf &
    -
    "T: " & .Controls(i).Top & " - " & D(i, 3) & vbCrLf &
    -
    "W: " & .Controls(i).Width & " - " & D(i, 0) &
    vbCrLf & _
    "H: " & .Controls(i).Height & " - " & D(i, 1) &
    vbCrLf

Done:
    If EchoOff Then DoCmd.Echo True
    End With

End Sub

Function ezGetScreenRes() As String
    'This function returns the windows screen size
    Dim R As RECT
    Dim hwnd As Long
    Dim RetVal As Long

    hwnd = GetDesktopWindow()
    RetVal = GetWindowRect(hwnd, R)
    ezGetScreenRes = (R.x2 - R.x1) & "x" & (R.y2 - R.y1)

End Function

Public Function ezGetScaleFactor(S) As Single
    'Returns a scale factor for resizing based on the passed
    parameter S
    'which should represent the screen size a form was designed
    for
    'the scale factor returned is based on the current screen
    resolution
    Select Case S
        Case 0 '640 x 480
            Select Case ezGetScreenRes
                Case "640x480"
                    ezGetScaleFactor = 1
                Case "800x600"
                    ezGetScaleFactor = 1.2
                Case "1024x768"
                    ezGetScaleFactor = 1.5
                Case "1152x864", "1152x870"
                    ezGetScaleFactor = 1.7
                Case "1280x1024"

```

```

        ezGetScaleFactor = 1.9
    Case "1600x1200"
        ezGetScaleFactor = 2.4
    End Select
Case -1 '800 x 600
    Select Case ezGetScreenRes
        Case "640x480"
            ezGetScaleFactor = 0.8
        Case "800x600"
            ezGetScaleFactor = 1
        Case "1024x768"
            ezGetScaleFactor = 1.2
        Case "1152x864", "1152x870"
            ezGetScaleFactor = 1.4
        Case "1280x1024"
            ezGetScaleFactor = 1.5
        Case "1600x1200"
            ezGetScaleFactor = 1.9
    End Select
Case -2 '1024 x 768
    Select Case ezGetScreenRes
        Case "640x480"
            ezGetScaleFactor = 0.6
        Case "800x600"
            ezGetScaleFactor = 0.7
        Case "1024x768"
            ezGetScaleFactor = 1
        Case "1152x864", "1152x870"
            ezGetScaleFactor = 1.05
        Case "1280x1024"
            ezGetScaleFactor = 1.1
        Case "1600x1200"
            ezGetScaleFactor = 1.4
    End Select
Case -3 '1280 x 1024
    Select Case ezGetScreenRes
        Case "640x480"
            ezGetScaleFactor = 0.5
        Case "800x600"
            ezGetScaleFactor = 0.6
        Case "1024x768"
            ezGetScaleFactor = 0.8
        Case "1152x864", "1152x870"
            ezGetScaleFactor = 0.9
        Case "1280x1024"
            ezGetScaleFactor = 1
        Case "1600x1200"
            ezGetScaleFactor = 1.1
    End Select
Case -4 '1600 x 1200
    Select Case ezGetScreenRes
        Case "640x480"
            ezGetScaleFactor = 0.3
        Case "800x600"
            ezGetScaleFactor = 0.4
        Case "1024x768"
            ezGetScaleFactor = 0.6
        Case "1152x864", "1152x870"
            ezGetScaleFactor = 0.65
        Case "1280x1024"
            ezGetScaleFactor = 0.7
        Case "1600x1200"
            ezGetScaleFactor = 1
    End Select
Case -5 '1152 x 864 OR 1152 x 870
    Select Case ezGetScreenRes
        Case "640x480"
            ezGetScaleFactor = 0.4

```

```

        Case "800x600"
            ezGetScaleFactor = 0.6
        Case "1024x768"
            ezGetScaleFactor = 0.8
        Case "1152x864", "1152x870"
            ezGetScaleFactor = 1
        Case "1280x1024"
            ezGetScaleFactor = 1.1
        Case "1600x1200"
            ezGetScaleFactor = 1.4
    End Select
    End Select
    If ezLargeFonts Then ezGetScaleFactor =
ezGetScaleFactor / 1.25
End Function
Public Function ezReSize(xForm As Form)
'This subroutine will resize the form based on it's current
dimensions
Dim ActiveForm As Object
Dim strTag As String
Dim SH As Single
Dim SW As Single

    On Error GoTo errorHandler
    Set ActiveForm = xForm

    'If form in datasheet view then don't resize
    If xForm.CurrentView <> 1 Then GoTo Done

    'If the form is maximized then don't resize
    If IsZoomed(xForm.hwnd) <> 0 Then GoTo Done

    'If the form is minimized then don't resize
    If IsIconic(xForm.hwnd) <> 0 Then GoTo Done

    With ActiveForm
        If .tag = "Sizing" Then GoTo Done
        strTag = .tag
        .tag = "Sizing"
        'Determine size of window and set resize based on
lowest proportion
        SH = .WindowHeight / .Section(0).Height
        SW = .WindowWidth / .Width
        If SH > SW Then
            ezSizeForm xForm, SW
        Else
            ezSizeForm xForm, SH
        End If
        .Width = 0
        On Error Resume Next
        .tag = strTag
    End With
    GoTo Done
errorHandler:
    MsgBox ERR.Description
Done:

End Function

Public Function ezLargeFonts() As Boolean
'This function returns a true if large fonts are being used.
Dim hdc As Long
Dim hwnd As Long
Dim PrevMapMode As Long
Dim tm As TEXTMETRIC

    'Get the handle of the desktop window
    hwnd = GetDesktopWindow()

```

```

'Get the device context for the desktop
hdc = GetWindowDC(hwnd)
If hdc Then 'Set the mapping mode to pixels
    PrevMapMode = SetMapMode(hdc, 1)
    'Get the size of the system font
    GetTextMetrics hdc, tm
    'Set the mapping mode back to what it was
    PrevMapMode = SetMapMode(hdc, PrevMapMode)
'Release the device context

```

```

ReleaseDC hwnd, hdc
'If the system font is more than 16 pixels high, then
large fonts are being used
    If tm.tmHeight > 16 Then ezLargeFonts = True Else
ezLargeFonts = False
    End If
End Function

```

## **MODULE-GlobalDeclaration**

```

Option Compare Database
Option Explicit
'#####
'      MODULE DESCRIPTION
'#####
'Class Name: GlobalDeclarations.bas
'
'Author: Pat Flanders & Scott Tufts
'
'Description: Contains all definitions for application global
'variables. Most of these are needed due to the inability of
'VBA to pass parameters as part of a constructor.
'
'References: None
'
'#####

'An object to represent the HFACs Connection file
Global oHFACSCConnection As HFACSCConnection
'Reusable object variable for the HFACSCConnection Class

'INI file declarations
Global gStrUID As String 'The user ID
Global gStrPWD As String 'The user password
Global gStrServerName As String 'The name of the MSDE
or SQL Server
Global gStrDatabaseFileName As String 'The name of the
mdf
Global gStrDatabaseName As String 'The name of the
database
Global gStrAppPath As String 'The application path
Global gStrAutoLogon As String 'Toggle for sa type login or
password
Global gStrFirstRun As String 'Toggle for determining if this
is the first time the program has been run.
Global gStrNTauth As String 'Toggle for determining if
NTAuth login should be attempted
Global gStrTypeDB As String 'The type of DB this program
will represent (mil, civ, or both).

'Security Settings
Global gBlnAdministrator As Boolean

'Value of the current connectionstring
Global gTheConnectionString As String

'Public Enums
Enum iTypeLogonConstants 'For logon prompts
    PROMPT = 1
    NOPROMPT = 2
End Enum

'The Operating System in use.
Global gStrOSType As String

'Program wide variables
Global gFormNeedsRefresh As Boolean 'Reusable flag for
identifying when a calling form needs to be refreshed when it
next gets the focus.

'Administration Variables
Global gLngMishapToGet As Long 'Reusable variable for
flagging a record
Global gBlnAddAMishap As Boolean 'Flag to identify that a
new record was added
Global gStrDescription As String 'For viewing of long
mishap descriptions on the select form.

'Query Variables
Global gStrInputString As String 'Reusable variable for input
string argument passing
Global bUseHFACSSummaryQuery As Boolean 'Flag for 2-
0-1-2-firm-ViewMishaps form to toggle which recordsource
to use.

'Graph Variables
Global gStrXFieldToGraph As String 'Name of X field for
Crosstab query under graph
Global gStrYFieldToGraph As String 'Name of Y field for
Crosstab query under graph

```

## APPENDIX G. CONNECTION COMPONENT

### CLASS-CallbackCls

Option Explicit

```
#####  
' CLASS DESCRIPTION  
#####  
'Class Name: CallbackCls.cls
```

'Author: Pat Flanders & Scott Tufts

'This class implements the cFTPCBK callback interface of the HFACS

'FTP server. The methods of this class provide the means for the

'HFACS server to notify (or callback) class instances from this

'component which utilize the FTP server functionality.

Basically,

'the members of this class provide a communication channel.

'ASIDE: The FTP server (HFACSFTP.exe) provides the functions needed

'to get FTP updates. These functions and their associated classes

'were removed from this component and compiled separately in order

'to work around the inability of Visual Basic to provide support

'for free threading. By placing the FTP functionality in a

'separately compiled executable, it can run in it's own process,

'which allows screen updates during long FTP downloads.

'References:

' - The HFACSFTP.exe ftp server.

'NOTE: See function headers for internal component references.

```
#####
```

Implements HFACSFTP.cFTPCBK 'Implement interface

```
*****  
' FUNCTIONS  
*****
```

```
=====
```

Function/Sub Name: cFTPCBK\_Complete()

'Description: An FTP update of the HFACS database requires the

'download of 2 files (HFACS.mdf & HFACS\_log.ldf). This function

'accepts messages from the the FTP server and notifies the 'frmFtpUpdate of progress. Specifically, of errors in

download

'and of successful download. If the first file is downloaded

'successfully (ErrCode = True And gIntCounter = 1), then this 'function notifies the frmFtpUpdate to begin the next

download.

'After successfully downloading both files, this function closes

'the frmFtpUpdate form.

'Input:

' ErrCode - Boolean value returned from FTP Server indicating

' success or failure of a file download.

'Output: None

'References:

' - The HFACSFTP.exe ftp server.

' - frmFtpUpdate.frm

' - HFACSMMain.bas

```
=====
```

##ModelId=3B294D27009C

Private Sub cFTPCBK\_Complete(ErrCode As Boolean)

'Determine if the first file was downloaded successfully  
If ErrCode = True And HFACSMMain.gIntCounter = 1

Then

frmFtpUpdate.GotFileDoNext

'Determine if the first file was downloaded successfully  
ElseIf ErrCode = True And HFACSMMain.gIntCounter = 2

Then

frmFtpUpdate.GotFileLast

'Either we are done or there was an error, so close  
frmFtpUpdate

Else

Unload frmFtpUpdate

End If

End Sub

## **CLASS-cErrorLog**

Option Explicit

#####

' CLASS DESCRIPTION

#####

'Class Name: cErrorLog.cls

'

'Author: Pat Flanders & Scott Tufts

'

'This writes stat us and error messages to the App.path

'ConnectionErrors.log file.

'

'References: None

'

'NOTE: See function headers for internal component references.

#####

\*\*\*\*\*

'

PROPERTIES

\*\*\*\*\*

'Integer value for each entry

Dim iErrorLog As Integer

\*\*\*\*\*

' DEFAULT NO-ARGUMENT CONSTRUCTOR  
(INITIALIZE EVENT)

\*\*\*\*\*

Private Sub Class\_Initialize()

iErrorLog = FreeFile

End Sub

\*\*\*\*\*

'

FUNCTIONS

\*\*\*\*\*

=====

=====

Function/Sub Name: ErrorLog()

'

'Description: Open the a file called ConnectionErrLog.log in the application path and write error etries to it.

'

'Input:

' strMsg - Message to write to the file

'

'Output: None

'

'References:

' - HFACSMMain.bas

=====

Public Sub ErrorLog(strMsg As String)

Debug.Print strMsg

Open HFACSMMain.gStrAppPath &

"ConnectionErrLog.log" For Append As iErrorLog

Print #iErrorLog, Now() & " : " & strMsg

Close iErrorLog

End Sub

=====

Function/Sub Name: ClearLog()

'

'Description: Clears the ConnectionErrLog.log.

'

'Input: None

'

'Output: None

'

'References:

' - HFACSMMain.bas

=====

Public Sub ClearLog()

Debug.Print "Error log cleared."

Open HFACSMMain.gStrAppPath &

"ConnectionErrLog.log" For Output As iErrorLog

Print #iErrorLog, Now() & " : " & "Log Cleared"

Close iErrorLog

End Sub

## **CLASS-HFACSCConnection**

Option Explicit

```
'#####
' CLASS DESCRIPTION
'#####
'Class Name: HFACSCConnection.cls
'
'Author: Pat Flanders & Scott Tufts
'
```

'This class is the controller class for the entire component. It  
'is the only class with public members accessible from  
'outside of the  
'component. Nothing can be manipulated without creating an  
'instance  
'of this class and using its methods to indirectly utilize the  
'functionality of the other classes.  
'

References:

```
' - Microsoft Data Formating Object Library 6.0
' - Microsoft ActiveX Data Objects 2.5 Library
' - Microsoft SQLDMO Object Library
' - Microsoft Scripting Runtime
' - GIF89 1.0 (For animated GIFs on Forms)
' - The HFACSFtp.exe ftp server.
'
```

'NOTE: See function headers for internal component  
references.

```
'#####
```

```
*****
'
' PROPERTYIES
'*****
```

'Variable for type logon (prompted or not-prompted)

```
##ModelId=3B294CF6035C
Private iTypeLogon As iTypeLogonConstants
```

'The user ID

```
##ModelId=3B294CF603D8
Private sUser As String
```

'The user password

```
##ModelId=3B294CF7003E
Private sPassword As String
```

'The name of the MSDE or SQL Server

```
##ModelId=3B294CF7008C
Private sSvrName As String
```

'The name of the .mdf file containing the database.

```
##ModelId=3B294CF700CB
Private sMDFName As String
```

'The name of the database

```
##ModelId=3B294CF70119
Private sDBName As String
```

'The application path

```
##ModelId=3B294CF70167
Private sInstDirectory As String
```

'Toggle to log on with/without prompt

```
##ModelId=3B294CF701A5
Private sAutomaticLogon As String
```

'Toggle for determining if this is the first run after an update.

```
##ModelId=3B294CF701F4
Private sFirstRunCheck As String
```

'Toggle for determining if NT authentication should be used  
for  
logon attempts.

```
##ModelId=3B294CF70242
Private sNTAuth As String
```

'The type of DB this program will represent (mil, civ, or  
both).

```
##ModelId=3B294CF70290
Private sTypeDB As String
```

'Variable to hold the value of the current connectionstring

```
##ModelId=3B294CF702DE
Private sTheConnectionString As String
```

'Enumerations for prompt/no-prompt functions

```
##ModelId=3B294CF60271
Public Enum iTypeLogonConstants
    ##ModelId=3B294CF6029F
    PROMPT = 1
    ##ModelId=3B294CF602DE
    NOPROMPT
End Enum
```

```
*****
'
' DEFAULT NO-ARGUMENT CONSTRUCTOR
' (INITIALIZE EVENT)
'*****
```

```
##ModelId=3B294CF7031C
Private Sub Class_Initialize()
```

'Set initial values for all variables by reading them from  
the

```
'HFACS.ini file.
Me.readINIFile
sUser = HFACSMMain.gStrUID
sPassword = HFACSMMain.gStrPWD
sSvrName = HFACSMMain.gStrServerName
sMDFName = HFACSMMain.gStrDatabaseFileName
sDBName = HFACSMMain.gStrDatabaseName
sInstDirectory = HFACSMMain.gStrAppPath
sAutomaticLogon = HFACSMMain.gStrAutoLogon
sFirstRunCheck = HFACSMMain.gStrFirstRun
sNTAuth = HFACSMMain.gStrNTauth
sTypeDB = HFACSMMain.gStrTypeDB
```

```
'Calculate a connection string
sTheConnectionString =
HFACSMMain.gTheConnectionString
```

```
'Clear the error log
Dim oTempClsErrorLog As New cErrorLog
oTempClsErrorLog.ClearLog
Set oTempClsErrorLog = Nothing
```

End Sub

```
*****
'
FUNCTIONS
*****
```

```
=====
'Function/Sub Name: Init()
'
'Description: If an instance of a class is created using the
pseudo-
constructors from the Constructors.bas module, this function
is
called to pass initial values, thereby mimicking the behavior
of
a constructor with arguments. Passed in values are all
required, but
the Constructors.New_HFACSCConnection() function
automatically sets
passed-in values to global variable values if they are left
blank.

'Input:
' sPassedInUser - The user ID
' sPassedInPassword - The user password
' sPassedInSvrName - The name of the MSDE or SQL
Server
' sPassedInMDFName - The name of the .mdf file
containing the
' database.
' sPassedInDBName - The name of the database
' sPassedInInstDirectory - The application path
' sPassedInAutomaticLogon - Toggle to log on with/without
prompt
' sPassedInFirstRunCheck - Toggle for determining if this
is the
' first run after an update.
' sPassedInNTAuth - Toggle for determining if NT
Auth.
' should be used for logon attempts.
' sPassedInTypeDB - The type of DB this program will
represent (mil, civ, or both).
'

'Output: None
'

'References:
' - Constructors.bas
' - HFACSMMain.bas
=====
'##ModelId=3B294CF7034B
Public Sub Init(sPassedInUser As String, sPassedInPassword
As String, sPassedInSvrName As String,
sPassedInMDFName As String, sPassedInDBName As
String, sPassedInInstDirectory As String,
sPassedInAutomaticLogon As String,
sPassedInFirstRunCheck As String,
sPassedInFirstRunAfterUpdate As String, sPassedInTypeDB
As String)

sUser = sPassedInUser
sPassword = sPassedInPassword
sSvrName = sPassedInSvrName
sMDFName = sPassedInMDFName
```

```
sDBName = sPassedInDBName
sInstDirectory = sPassedInInstDirectory
sAutomaticLogon = sPassedInAutomaticLogon
sFirstRunCheck = sPassedInFirstRunCheck
sNTAuth = sPassedInFirstRunAfterUpdate
sTypeDB = sPassedInTypeDB
```

End Sub

```
=====
'Function/Sub Name: doConnect()
'
'Description: This procedure will make a connection to a
database
server based on the value of iTypeLogonIn. If this
parameter is
left blank, the class determines the appropriate type of logon
to perform. This function also detects if it is the first time
HFACS has been run and displays the frmWelcome.frm as
appropriate.
After a successful logon, it sets the .ini value indicating a
first run to "F."
'

'Input:
' iTypeLogonIn - Type of logon to perform (prompted or
not-prompted.
'

'Output: Logon success or failure.
'

'References:
' - frmODBLogon.frm
' - frmWelcome.frm
' - MSDE.cls
' - INIFileController.cls
' - Constructors.bas
' - HFACSMMain.bas
=====
'##ModelId=3B294CF8007D
Public Function doConnect(Optional iTypeLogonIn As
iTypeLogonConstants) As Boolean

On Error GoTo StartError
'Check for optional arguments and assign to defaults as
needed.
If iTypeLogonIn = 0 Then
'A no-prompt logon can only be made on the local
machine
'and if no password is needed.
If sAutomaticLogon = "T" And sPassword = "" And _
sSvrName = "(local)" Then
iTypeLogon = NOPROMPT
Else
iTypeLogon = PROMPT
End If
Else
iTypeLogon = iTypeLogonIn
End If

'Variables for testing success or failure of various
operations
Dim bConstructorSuccess As Boolean
Dim bTestSuccess As Boolean

Select Case iTypeLogon

*****
Case PROMPT 'Prompt logon
```



```

        'If this is a first run, show the welcome form.
        If HFACSMMain.gStrFirstRun = "T" Then
            frmWelcome.Show 1
        End If

        frmODBLogon.Show 1 'Show the logon form

        'Test for successful logon
        If gblnPromptedLogonSuccess = True Then _
            doConnect = True Else doConnect = False

        'If this was a successful first run reset the first
        'run flag. This should never change again.
        If gblnPromptedLogonSuccess = True And _
            HFACSMMain.gStrFirstRun = "T" Then
            gStrFirstRun = "F"
        End If

        *****

        Case NOPROMPT 'No Prompt logon

            'Create an instance of MSDE
            bConstructorSuccess =
Constructors.New_MSDE(sUser, _
                sPassword, sSvrName, sMDFName, sDBName,
            _
                sInstDirectory, sAutomaticLogon,
sFirstRunCheck, _
                sNTAuth, sTypeDB)

            'Start the server and copy the database to it, if
            'needed.
            bTestSuccess = oMSDE.StartAndCopy

            'Test for success
            If bTestSuccess Then doConnect = True _
                Else doConnect = False
            Set oMSDE = Nothing

            'If this was a successful first run reset the first
            'run flag. This should never change again.
            If bTestSuccess = True And
HFACSMMain.gStrFirstRun _
                = "T" Then
                gStrFirstRun = "F"
            End If

        *****

        Case Else ' Default to a an error message, something is
        wrong.
            MsgBox "Can't determine how to connect." & _
                " Contact your system administrator.", _
                vbCritical + vbOKOnly, "Error"
            doConnect = False

        *****

        End Select

    ExitSub:

        'Update the global connection string
        createConnectionString

        'If doConnect() was a success, save all the settings
        'and so the Access .adp has knows what transpired.
        If doConnect = True Then
            Constructors.New_INIFileController

                oINIFileController.writeINIentries
                Set oINIFileController = Nothing
            End If

            Exit Function

        StartError:
            MsgBox "Error making a connection to HFACS." &
Chr(13) & _
                Chr(13) & "The detailed error message is: " & _
                    Err.Description & Chr(13) & Chr(13) & "Error Number:
" & _
                    Err.Number
            doConnect = False
            Resume ExitSub
            Resume Next
        End Function

        '=====
        'Function/Sub Name: createConnectionString()
        '
        'Description: This procedure updates the value of
        'the global variable for the connection string that will be used
        'for
        'all ADO connections (hfacsmain.gTheConnectionString). It
        'determines
        'if the string should use NT authentication or regular SQL
        'based on the global variable gStrNTauth.
        '
        'Input: None
        '
        'Output: success or failure of update.
        '
        'References:
        ' - Constructors.bas
        ' - HFACSMMain.bas
        '=====
        '##ModelId=3B294CF800BB
        Private Function createConnectionString() As Boolean

            On Error GoTo StartError
            Screen.MousePointer = 11

            'Determine which type of string to create
            If HFACSMMain.gStrNTauth = "T" Then
                gTheConnectionString =
"PROVIDER=SQLOLEDB.1;INTEGRATED" & _
                    " SECURITY=SSPI;PERSIST SECURITY
INFO=FALSE;INITIAL CATALOG=" & _
                        gStrDatabaseName & ";DATA SOURCE=" &
gStrServerName
                Else
                    gTheConnectionString =
"PROVIDER=SQLOLEDB.1;PASSWORD=" & _
                        gStrPWD & ";PERSIST SECURITY
INFO=TRUE;USER ID=" & _
                            gStrUID & ";INITIAL CATALOG=" &
gStrDatabaseName & _
                                ";DATA SOURCE=" & gStrServerName
                End If
            Screen.MousePointer = 0
            createConnectionString = True

        ExitSub:
            Exit Function

        StartError:

```

```

Screen.MousePointer = 0
createConnectionString = False
Resume ExitSub

End Function

'=====
'Function/Sub Name: getUpdateFTP()
'
'Description: This function creates an instance of the
'UpdateController class, providing access to FTP updates.
'
'Input: None
'
'Output: success or failure of update.
'
'References:
' - Constructors.bas
' - UpdateController.cls
' - HFACSMMain.bas
'=====

##ModelId=3B294CF800EA
Public Function getUpdateFTP() As Boolean

    On Error GoTo StartError

    'Open the FTP form by creating an UpdateController
    object
    Set HFACSMMain.oUpdateController = New
    UpdateController
    getUpdateFTP =
    HFACSMMain.oUpdateController.getUpdate

ExitSub:
    Destroy it when done
    Set HFACSMMain.oUpdateController = Nothing
    Exit Function

StartError:
    getUpdateFTP = False
    Resume ExitSub

End Function

'=====
'Function/Sub Name: getUpdateDisk()
'
'Description: This function creates an instance of the
'UpdateController class, providing access to update from disk
'functionality.
'
'Input: None
'
'Output: success or failure of update.
'
'References:
' - Constructors.bas
' - UpdateController.cls
' - HFACSMMain.bas
'=====

##ModelId=3B294CF80119
Public Function getUpdateDisk() As Boolean

    On Error GoTo StartError

```

```

'Open the File Open dialog by creating an
UpdateController object
    Set HFACSMMain.oUpdateController = New
    UpdateController
    getUpdateDisk =
    HFACSMMain.oUpdateController.getUpdateDisk

ExitSub:
    Destroy it when done
    Set HFACSMMain.oUpdateController = Nothing
    Exit Function

StartError:
    getUpdateDisk = False
    Resume ExitSub

End Function

'=====
'Function/Sub Name: writeINIFile()
'
'Description: This function creates an instance of the
'INIFileController class, providing methods to write to the
'HFACS.ini
'file.
'
'Input: None
'
'Output: success or failure of write.
'
'References:
' - Constructors.bas
' - INIFileController.cls
' - HFACSMMain.bas
'=====

##ModelId=3B294CF80138
Public Function writeINIFile() As Boolean

    On Error GoTo StartError

    'Open and write to HFACS.ini by creating an
    UpdateController
    object.
    Set HFACSMMain.oINIFileController = New
    INIFileController
    writeINIFile =
    HFACSMMain.oINIFileController.writeINIentries

ExitSub:
    Destroy it when done
    Set HFACSMMain.oINIFileController = Nothing
    Exit Function

StartError:
    writeINIFile = False
    Resume ExitSub

End Function

'=====
'Function/Sub Name: readINIFile()
'
'Description: This function creates an instance of the
'INIFileController class, providing methods to read from the
'HFACS.ini
'file.

```

```

'
'Input: None
'
'Output: success or failure of read.
'
'References:
' - Constructors.bas
' - INIFileController.cls
' - HFACSMMain.bas
'=====
'##ModelId=3B294CF80167
Public Function readINIFile() As Boolean

    On Error GoTo StartError

    'Open and read HFACS.ini by creating an
UpdateController
    'object.
    Set HFACSMMain.oINIFileController = New
INIFileController
    readINIFile =
HFACSMMain.oINIFileController.readINIEntries

ExitSub:
    Destroy it when done
    Set HFACSMMain.oINIFileController = Nothing
    Exit Function

StartError:
    readINIFile = False
    Resume ExitSub

End Function

'=====
'Function/Sub Name: getSQLServerPath()
'
'Description: This function gets the path to the SQL server.
'
'Input: None
'
'Output: String value of the SQL server.
'
'References:
' - HFACSMMain.bas
'=====
'##ModelId=3B294CF80167
Public Function getSQLServerPath() As String

    getSQLServerPath = HFACSMMain.gSQLServerPath

End Function

'*****
'
' Public Property GET and LET statements follow
'
'*****

'##ModelId=3B294CF801F4
Public Property Get User() As Variant
    User = gStrUID
End Property
'##ModelId=3B294CF80280
Public Property Get Password() As Variant

```

```

    Password = gStrPWD
End Property
'##ModelId=3B294CF8031C
Public Property Get ServerName() As Variant
    ServerName = gStrServerName
End Property
'##ModelId=3B294CF803A9
Public Property Get DatabaseFileName() As Variant
    DatabaseFileName = gStrDatabaseFileName
End Property
'##ModelId=3B294CF9005D
Public Property Get DatabaseName() As Variant
    DatabaseName = gStrDatabaseName
End Property
'##ModelId=3B294CF900EA
Public Property Get AppPath() As Variant
    AppPath = gStrAppPath
End Property
'##ModelId=3B294CF90186
Public Property Get AutomaticLogon() As Variant
    AutomaticLogon = gStrAutoLogon
End Property
'##ModelId=3B294CF90213
Public Property Get FirstRunCheck() As Variant
    FirstRunCheck = gStrFirstRun
End Property
'##ModelId=3B294CF902AF
Public Property Get UseNTAuth() As Variant
    UseNTAuth = gStrNTauth
End Property
'##ModelId=3B294CF9033C
Public Property Get TypeDatabase() As Variant
    TypeDatabase = gStrTypeDB
End Property
'##ModelId=3B294CF9037A
Public Property Get ConnectionString() As Variant
    ConnectionString = gTheConnectionString
End Property

'Property LET Statements
'##ModelId=3B294CF80196
Public Property Let User(ByVal vNewValue As Variant)
    sUser = vNewValue
    HFACSMMain.gStrUID = vNewValue
End Property
'##ModelId=3B294CF80222
Public Property Let Password(ByVal vNewValue As
Variant)
    sPassword = vNewValue
    HFACSMMain.gStrPWD = vNewValue
End Property
'##ModelId=3B294CF802BF
Public Property Let ServerName(ByVal vNewValue As
Variant)
    sSvrName = vNewValue
    HFACSMMain.gStrServerName = vNewValue
End Property
'##ModelId=3B294CF8034B
Public Property Let DatabaseFileName(ByVal vNewValue
As Variant)
    sMDFName = vNewValue
    HFACSMMain.gStrDatabaseFileName = vNewValue
End Property
'##ModelId=3B294CF90000
Public Property Let DatabaseName(ByVal vNewValue As
Variant)
    sDBName = vNewValue
    HFACSMMain.gStrDatabaseName = vNewValue

```

```

End Property
##ModelId=3B294CF9008C
Public Property Let AppPath(ByVal vNewValue As Variant)
    sInstDirectory = vNewValue
    HFACSMMain.gStrAppPath = vNewValue
End Property
##ModelId=3B294CF90128
Public Property Let AutomaticLogon(ByVal vNewValue As
Variant)
    sAutomaticLogon = vNewValue
    HFACSMMain.gStrAutoLogon = vNewValue
End Property
##ModelId=3B294CF901B5
Public Property Let FirstRunCheck(ByVal vNewValue As
Variant)

```

```

    sFirstRunCheck = vNewValue
    HFACSMMain.gStrFirstRun = vNewValue
End Property
##ModelId=3B294CF90251
Public Property Let UseNTAuth(ByVal vNewValue As
Variant)
    sNTAuth = vNewValue
    HFACSMMain.gStrNTauth = vNewValue
End Property
##ModelId=3B294CF902DE
Public Property Let TypeDatabase(ByVal vNewValue As
Variant)
    sTypeDB = vNewValue
    HFACSMMain.gStrTypeDB = vNewValue
End Property

```

## CLASS-INIFile

Option Explicit

```
'#####  
' CLASS DESCRIPTION  
'#####  
'Class Name: INIFile.cls  
'  
'Author: Microsoft Corporation. Modified by Pat Flanders  
&  
' Scott Tufts  
'  
'This class creates .ini File objects used to create, delete, set,  
'and get values in a standard format Microsoft .ini file. It  
'uses  
'calls to the Windows API for efficiency.  
'
```

'References: Windows API

'NOTE: See function headers for internal component references.

```
'#####  
  
'*****  
' PROPERTIES  
'*****
```

'The name of the ini file to read

```
'##ModelId=3B294CFD03A9  
Private msWbkName As String
```

'API Wrapper Code - provided by Microsoft

```
'##ModelId=3B294CFE0000  
Private Declare Function WritePrivateProfileString Lib  
"kernel32" Alias "WritePrivateProfileStringA" (ByVal  
lpApplicationName As String, ByVal lpKeyName As String,  
ByVal lpString As String, ByVal lpFileName As String) As Long
```

```
'##ModelId=3B294CFE00AB  
Private Declare Function GetPrivateProfileString Lib  
"kernel32" Alias "GetPrivateProfileStringA" (ByVal  
lpApplicationName As String, ByVal lpKeyName As Any,  
ByVal lpDefault As String, ByVal lpReturnedString As  
String, ByVal nSize As Long, ByVal lpFileName As String)  
As Long
```

```
'##ModelId=3B294CFE0196  
Private Declare Function GetWindowsDirectory Lib  
"kernel32" Alias "GetWindowsDirectoryA" (ByVal lpBuffer  
As String, ByVal nSize As Long) As Long
```

```
'*****  
' FUNCTIONS  
'*****
```

```
'=====  
'Function/Sub Name: Init()  
'
```

'Description: If an instance of a class is created using the  
psuedo-  
'constructors from the Constructors.bas module, this function  
is  
'called to pass initial values, thereby mimicking the behavior  
of  
'a constructor with arguments. Passed in values are all  
required, but  
'the Constructors.New\_INIFile() function automatically sets  
'passed-in values to global variable values if they are left  
'blank.  
'

'Input:  
' sPassedInWorkBookName - Name of the .ini file to  
manipulate  
'

'Output: None

'References:  
' - Constructors.bas

```
'=====  
##ModelId=3B294CFE0213  
Friend Sub Init(sPassedInWorkBookName As String)
```

msWbkName = sPassedInWorkBookName

End Sub

```
'=====  
'Function/Sub Name: WriteToIniFile()  
'
```

'Description: Write a section, key, and value to an .ini file.

'Input:  
' strSection - Name of a section  
' strKey - Name of a key  
' strValue - Name of a key value  
' strFileName - Name of the file to manipulate  
'

'Output: Success or failure

'References: None

```
'=====  
##ModelId=3B294CFE0251  
Friend Function WriteToIniFile(strSection As String, strKey  
As String, strValue As String, strFileName As String) As  
Boolean
```

' Pass in name of section, key, key value, and file name.  
If WritePrivateProfileString(strSection, strKey, \_  
strValue, strFileName) Then  
WriteToIniFile = True  
Else  
MsgBox "Error writing to .ini file: " & Err.LastDllError  
WriteToIniFile = False  
End If

End Function

```
'=====  
'Function/Sub Name: DeleteIniSection()  
'
```

'Description: Delete a section and all of its keys from an .ini file.

'Input:

' strSection - Name of a section  
' strFileName - Name of the file to manipulate

'Output: Success or failure

'References: None

=====  
##ModelId=3B294CFE02DE

Friend Function DeleteIniSection(strSection As String, strFileName As String) As Boolean

```
If WritePrivateProfileString(strSection, vbNullString, _
    vbNullString, strFileName) Then
    DeleteIniSection = True
Else
    MsgBox "Error deleting section from .ini file: " _
        & Err.LastDllError
    DeleteIniSection = False
End If
```

End Function

=====  
Function/Sub Name: DeleteIniKey()

'Description: Delete a key and its value from an .ini file.

'Input:

' strSection - Name of a section  
' strKey - Name of a key  
' strFileName - Name of the file to manipulate

'Output: Success or failure

'References: None

=====  
##ModelId=3B294CFE033C

Friend Function DeleteIniKey(strSection As String, strKey As String, strFileName As String) As Boolean

```
If WritePrivateProfileString(strSection, strKey, _
    vbNullString, strFileName) Then
    DeleteIniKey = True
Else
    MsgBox "Error deleting section from .ini file: " _
        & Err.LastDllError
    DeleteIniKey = False
End If
```

End Function

=====  
Function/Sub Name: GetIniFileName()

'Description: Return name for .ini file. Name includes name of workbook file and ".ini". File path can be made the Windows directory.  
'by uncommenting the code below

'Input: None

'Output: String path (e.g. C:\windows\HFACS.ini).

'References: None

=====  
##ModelId=3B294CFE03A9

Friend Function GetIniFileName() As String

```
Dim strWinDir As String
Dim lngLen As Long
```

```
' Create null-terminated string to pass to
' GetWindowsDirectory.
strWinDir = String$(255, vbNullChar)
```

```
lngLen = Len(strWinDir)
```

```
' Return Windows directory.
GetWindowsDirectory strWinDir, lngLen
```

```
' Truncate before first null character.
strWinDir = Left(strWinDir, _
    InStr(strWinDir, vbNullChar) - 1)
```

```
' Return .ini file name.
GetIniFileName = strWinDir & "\" & msWbkName &
".ini"
```

GetIniFileName = App.Path & "\" & msWbkName & ".ini"

End Function

=====  
Function/Sub Name: ReadFromIniFile()

'Description: Read a value from an .ini file, given the file name, section, key, and default value to return if key is not found.

'Input:

' strSection - Name of a section  
' strKey - Name of a key  
' strDefault - Default name of a key value  
' strFileName - Name of the file to manipulate

'Output: Success or failure

'References: None

=====  
##ModelId=3B294CFE03D8

Friend Function ReadFromIniFile(strFileName As String, strSection As String, strKey As String, Optional strDefault As String = "") As String

```
Dim strValue As String
```

```
' Fill string buffer with null characters.
strValue = String$(255, vbNullChar)
```

```
' Attempt to read value. GetPrivateProfileString
' function returns number of characters written
' into string.
```

```
If GetPrivateProfileString(strSection, strKey, _
    strDefault, strValue, Len(strValue), _
    strFileName) > 0 Then
    ' If characters have been written into string, parse string
    ' and return.
```

<pre> 1) strValue = Left(strValue, InStr(strValue, vbNullChar) -     ReadFromIniFile = strValue Else     ' Otherwise, return a zero-length string. </pre>	<pre>         ReadFromIniFile = strDefault     End If End Function </pre>
---	---

## **CLASS-INIFileController**

Option Explicit

```
#####
' CLASS DESCRIPTION
#####
'Class Name: INIFileController.cls
'
'Author: Pat Flanders & Scott Tufts
'
'
'This class creates instances of INIFile.cls used to create,
delete,
'set, and get values in a standard format Microsoft .ini file.
'
'References: None
'
'NOTE: See function headers for internal component
references.
'#####
```

```
*****
'
' FUNCTION S
'
'*****
```

```
=====
'Function/Sub Name: Init()
'
'Description: If an instance of a class is created using the
psuedo-
'constructors from the Constructors.bas module, this function
is
'called to pass initial values, thereby mimicking the behavior
of
'a constructor with arguments. Passed in values are all
required, but
'the Constructors.New_INIFileController() function
automatically sets
'passed-in values to global variable values if they are left
'blank.
'
'Input: None
'
'Output: None
'
'References: None
'=====
'##ModelId=3B294D0C01A5
Friend Sub Init()
```

'Do nothing. This function body is provided for future use.

End Sub

```
=====
'Function/Sub Name: readINIentries()
'
'Description: This function creates an instance of the
INIFile class and reads values from the HFACS.ini file.
'
'Input:
```

```
' sFileName - The name of the .ini file to read
'
'Output: success or failure of read.
'
'References:
' - Constructors.bas
' - INIFile.cls
' - HFACSMMain.bas
'=====
'##ModelId=3B294D0C01D4
Friend Function readINIentries(Optional sFileName As
String) As Boolean

'Set the MSDE class instance default values
If IsMissing(sFileName) Then sFileName =
gINIFILENAME

On Error GoTo StartError
Screen.MousePointer = 11
Debug.Print "Reading ini data . . ."

'Create oINIFile
Constructors.New_INIFile sFileName

' Get name for .ini file in the SYSTEM directory
gStrFileName = oINIFile.GetIniFileName

' Read values from .ini file. Specify file name, section, and
key.
gStrUID = oINIFile.ReadFromIniFile(gStrFileName, _
"CONNECTION", "UID")
gStrPWD = oINIFile.ReadFromIniFile(gStrFileName, _
"CONNECTION", "PWD")
gStrServerName =
oINIFile.ReadFromIniFile(gStrFileName, _
"CONNECTION", "ServerName")
gStrDatabaseFileName =
oINIFile.ReadFromIniFile(gStrFileName, _
"CONNECTION", "DatabaseFileName")
gStrDatabaseName =
oINIFile.ReadFromIniFile(gStrFileName, _
"CONNECTION", "DatabaseName")
gStrAppPath = oINIFile.ReadFromIniFile(gStrFileName,
-
"CONNECTION", "InstallDir")
gStrAutoLogon =
oINIFile.ReadFromIniFile(gStrFileName, _
"CONNECTION", "AutoLogon")
gStrFirstRun = oINIFile.ReadFromIniFile(gStrFileName,
-
"CONNECTION", "FirstRun")
gStrNTauth = oINIFile.ReadFromIniFile(gStrFileName, _
"CONNECTION", "NTAuth")
gStrTypeDB = oINIFile.ReadFromIniFile(gStrFileName, _
"DBTYPE", "DBtype")

Screen.MousePointer = 0
readINIentries = True

ExitSub:
Set oINIFile = Nothing
Exit Function

StartError:
```



```

Screen.MousePointer = 0
readINIentries = False
Resume ExitSub

End Function

'=====
'Function/Sub Name: writeINIentries()
'
'Description: This function creates an instance of the
'INIFile class and writes values to the HFACS.ini file.
'
'Input:
' sUser      - The user ID
' sPassword   - The user password
' sSvrName    - The name of the MSDE or SQL Server
' sMDFName    - The name of the .mdf file containing the
'              database.
' sDBName     - The name of the database
' sInstDirectory - The application path
' sAutomaticLogon - Toggle to log on with/without prompt
' sFirstRunCheck - Toggle for determining if this is the
'                  first run after an update.
' sNTAuth     - Toggle for determining if NT Auth.
'              should be used for logon attempts.
' sTypeDB     - The type of DB this program will
'              represent (mil, civ, or both).'
'
'Output: success or failure of write.
'
'References:
' - Constructors.bas
' - INIFile.cls
' - HFACSMMain.bas
'=====

##ModelId=3B294D0C0222
Friend Function writeINIentries(Optional sUser As String,
Optional sPassword As String, Optional sSvrName As String,
Optional sMDFName As String, Optional sDBName As
String, Optional sInstDirectory As String, Optional
sAutomaticLogon As String, Optional sFirstRunCheck As
String, Optional sNTAuth As String, Optional sTypeDB As
String) As Boolean

    On Error GoTo StartError
    Screen.MousePointer = 11
    Debug.Print "Writing ini data . . ."

    'Create oINIFile
    Constructors.New_INIFile gINIFILENAME

    'Check for optional arguments and assign to defaults as
    needed.
    If Trim(sSvrName) = "" Then
        sSvrName = gStrServerName
    End If
    If Trim(sUser) = "" Then
        sUser = gStrUID
    End If
    If Trim(sPassword) = "" Then
        sPassword = gStrPWD
    End If
    If Trim(sMDFName) = "" Then
        sMDFName = gStrDatabaseFileName
    End If
    If Trim(sDBName) = "" Then
        sDBName = gStrDatabaseName

    End If
    If Trim(sInstDirectory) = "" Then
        sInstDirectory = gStrAppPath
    End If
    If Trim(sAutomaticLogon) = "" Then
        sAutomaticLogon = gStrAutoLogon
    End If
    If Trim(sFirstRunCheck) = "" Then
        sFirstRunCheck = gStrFirstRun
    End If
    If Trim(sNTAuth) = "" Then
        sNTAuth = gStrNTAuth
    End If
    If Trim(sTypeDB) = "" Then
        sTypeDB = gStrTypeDB
    End If

    'Remove this block to allow updating of passwords in the .ini
    file
    *****
    'If the user is using an account other than on the local
    server,
    'then it will always require a password.
    'Passwords can't be stored in the clear (like in the .ini file),
    'so never update them.

    If sPassword = "" And sSvrName = "(local)" _
        And sFirstRunCheck = "F" Then
        sAutomaticLogon = "T"
    Else
        sAutomaticLogon = "F"

        'Update the value of the global variable for password
        now.
        gStrPWD = sPassword

        'Now set the local value to "" so it doesn't get written
        "in the ini file"
        sPassword = ""

    End If

    *****

    Dim writeSuccess As Boolean
    'Write the new values to the .ini file
    writeSuccess =
oINIFile.WriteToIniFile("CONNECTION", _
    "UID", sUser, gStrFileName)
    writeSuccess =
oINIFile.WriteToIniFile("CONNECTION", _
    "PWD", sPassword, gStrFileName)
    writeSuccess =
oINIFile.WriteToIniFile("CONNECTION", _
    "ServerName", sSvrName, gStrFileName)
    writeSuccess =
oINIFile.WriteToIniFile("CONNECTION", _
    "DatabaseFileName", sMDFName, gStrFileName)
    writeSuccess =
oINIFile.WriteToIniFile("CONNECTION", _
    "DatabaseName", sDBName, gStrFileName)
    writeSuccess =
oINIFile.WriteToIniFile("CONNECTION", _
    "InstallDir", sInstDirectory, gStrFileName)
    writeSuccess =
oINIFile.WriteToIniFile("CONNECTION", _
    "AutoLogon", sAutomaticLogon, gStrFileName)

```

```

writeSuccess =
oINIFile.WriteToIniFile("CONNECTION", _
    "FirstRun", sFirstRunCheck, gStrFileName)
writeSuccess =
oINIFile.WriteToIniFile("CONNECTION", _
    "NTAuth", sNTAuth, gStrFileName)
writeSuccess = oINIFile.WriteToIniFile("DBTYPE", _
    "DBtype", sTypeDB, gStrFileName)

'Update global variables to the new values
gStrUID = sUser

'**** Un-comment this to allow updating of passwords
'in the .ini file.
'gStrPWD = sPassword

gStrServerName = sSvrName
gStrDatabaseFileName = sMDFName
gStrDatabaseName = sDBName
gStrAppPath = sInstDirectory

```

```

gStrAutoLogon = sAutomaticLogon
gStrFirstRun = sFirstRunCheck
gStrNTauth = sNTAuth
gStrTypeDB = sTypeDB

writeINentries = True

ExitSub:
    Set oINIFile = Nothing
    Screen.MousePointer = 0
    Exit Function

StartError:
    Screen.MousePointer = 0
    writeINentries = False
    Resume ExitSub

End Function

```

## **CLASS-MSDE**

Option Explicit

```
'#####  
' CLASS DESCRIPTION  
'#####  
'Class Name: MSDE.cls  
'  
'Author: Pat Flanders & Scott Tufts  
'  
'This class is responsible for starting the MSDE or SQL  
server, ensuring  
'that the HFACS database is installed, and managing database  
updates.  
'  
'References:  
' - Microsoft Data Formating Object Library 6.0  
' - Microsoft ActiveX Data Objects 2.5 Library  
' - Microsoft SQLDMO Object Library  
' - Microsoft Scripting Runtime  
'  
'NOTE: See function headers for internal component  
references.  
'#####
```

```
*****  
' PROP ERTIES  
*****
```

The user ID  
##ModelId=3B294D2201D4  
Private sUser As String

The user password  
##ModelId=3B294D220222  
Private sPassword As String

The name of the MSDE or SQL Server  
##ModelId=3B294D220261  
Private sSvrName As String

The name of the .mdf file containing the database.  
##ModelId=3B294D22029F  
Private sMDFName As String

The name of the database  
##ModelId=3B294D2202EE  
Private sDBName As String

The application path  
##ModelId=3B294D22032C  
Private sInstDirectory As String

Toggle to log on with/without prompt  
##ModelId=3B294D22037A  
Private sAutomaticLogon As String

Toggle for determining if this is the first run after an update.  
##ModelId=3B294D2203B9  
Private sFirstRunCheck As String

'Toggle for determining if NT authentication should be used  
for

'logon attempts.  
##ModelId=3B294D23001F  
Private sNTAuth As String

'The type of DB this program will represent (mil, civ, or  
both).

##ModelId=3B294D23005D  
Private sTypeDB As String

'Variable for writing to the errorlog  
Private oClsErrorLog As cErrorLog

```
*****  
' DEFAULT NO-ARGUMENT CONSTRUCTOR  
(INITIALIZE EVENT)  
*****  
##ModelId=3B294D23009C  
Private Sub Class_Initialize()
```

Set oClsErrorLog = New cErrorLog

sUser = gStrUID  
sPassword = gStrPWD  
sSvrName = gStrServerName  
sMDFName = gStrDatabaseFileName  
sDBName = gStrDatabaseName  
sInstDirectory = gStrAppPath  
sAutomaticLogon = gStrAutoLogon  
sFirstRunCheck = gStrFirstRun  
sNTAuth = gStrNTAuth  
sTypeDB = gStrTypeDB

End Sub

```
*****  
' FUNCTIONS  
*****
```

=====

'Function/Sub Name: Init()

'Description: If an instance of a class is created using the  
psuedo-  
'constructors from the Constructors.bas module, this function  
is  
'called to pass initial values, thereby mimicking the behavior  
of  
'a constructor with arguments. Passed in values are all  
required, but  
'the Constructors.New\_MSDE() function automatically sets  
'passed-in values to global variable values if they are left  
'blank.

'Input:  
' sPassedInUser - The user ID  
' sPassedInPassword - The user password

```

' sPassedInSvrName      - The name of the MSDE or
SQL Server
' sPassedInMDFName      - The name of the .mdf file
containing the
'                        database.
' sPassedInDBName       - The name of the database
' sPassedInInstDirectory - The application path
' sPassedInAutomaticLogon - Toggle to log on
with/without prompt
' sPassedInFirstRunCheck - Toggle for determining if
this is the
'                        first run after an update.
' sPassedInFirstRunAfterUpdate - Toggle for determining if
NT Auth.
'                        should be used for logon attempts.
' sPassedInTypeDB       - The type of DB this program
will
'                        represent (mil, civ, or both).'

'Output: None

'References:
' - Constructors.bas
' - HFACSMMain.bas
'=====

'##ModelId=3B294D2300CB
Friend Sub Init(sPassedInUser As String, sPassedInPassword
As String, sPassedInSvrName As String,
sPassedInMDFName As String, sPassedInDBName As
String, sPassedInInstDirectory As String,
sPassedInAutomaticLogon As String,
sPassedInFirstRunCheck As String,
sPassedInFirstRunAfterUpdate As String, sPassedInTypeDB
As String)

    sUser = sPassedInUser
    sPassword = sPassedInPassword
    sSvrName = sPassedInSvrName
    sMDFName = sPassedInMDFName
    sDBName = sPassedInDBName
    sInstDirectory = sPassedInInstDirectory
    sAutomaticLogon = sPassedInAutomaticLogon
    sFirstRunCheck = sPassedInFirstRunCheck
    sNTAuth = sPassedInFirstRunAfterUpdate
    sTypeDB = sPassedInTypeDB

End Sub

'=====
'Function/Sub Name: startMSDE()
'
'Description: This procedure will start an instance SQL
Server and
'create a connection to it, thereby verifying that the specified
'server exists and that it is started. If the server is already
running,
'the error trap will exit the procedure and leave the server
running.
'
'A bug in SQL Server 2000 prevents SQLDMO from starting a
remote server
'so this code also detects the error and switches to an ADO
type
'connection to verify that the HFACS database is present on
the remote
'machine. In the case of the ADO connection, a copy the
database

```

```

'either exists or doesn't exist on the remote server. If the
ADO
'connection fails, a global flag is set so that all classes
'in the component know NOT to try to copy an instance of
the database
'to the remote server, which would generate another error.
'
'Input:
' sSvrNameIn  The server to be started
' sUserIn     The user ID with which to start the server
' sPasswordIn The user password
'
'Output: Success or Failure
'
'References:
' - Constructors.bas
' - HFACSMMain.bas
'=====

'##ModelId=3B294D2301D4
Friend Function startMSDE(Optional sSvrNameIn As String,
Optional sUserIn As String, Optional sPasswordIn As String)
As Boolean

    Screen.MousePointer = 11
    On Error GoTo StartError

    'Check for optional arguments and assign to defaults as
needed.
    If Trim(sSvrNameIn) <> "" Then
        sSvrName = sSvrNameIn
    End If
    If Trim(sUserIn) <> "" Then
        sUser = sUserIn
    End If
    If Trim(sPasswordIn) <> "" Then
        sPassword = sPasswordIn
    End If

    Dim iSlowServerCounter As Integer
    iSlowServerCounter = 1

    'Only use SQLDMO for local machine operations
    Dim iADOAttemptCounter As Integer
    iADOAttemptCounter = 0
    If Trim(sSvrName) <> "(local)" Then
        Err.Raise -2147221163
    End If

    'Declare an object for SQL server manipulation
    Dim osvrl As sqlldmo.SQLServer

    'Create the SQLDMO Server Object.
    Set osvrl = CreateObject("SQLDMO.sqlserver")

    osvrl.LoginTimeout = 20
    'Start Server.

    'Reset the no copy needed variable to false every time
'connection is attempted.
    gblnNoCopyNeeded = False

TimeoutResume:
    'Determine connection type
    If sNTAuth = "T" Then
        osvrl.LoginSecure = True
    End If
    oClsErrorLog.ErrorLog "startMSDE-Attempting to start
server. . ."

```

```

osvr.Start True, sSvrName, sUser, sPassword
oClsErrorLog.ErrorLog "startMSDE-The server was
successfully started . . ."

StartConnect:
oClsErrorLog.ErrorLog "startMSDE-Attempting to
connect . . ."
'Attempt a connection.
'If no login name, use NT Integrated security in an attempt.
'to connect.
If sNTAuth = "T" Then
osvr.LoginSecure = True
End If
'This is the actual connection attempt.
osvr.Connect sSvrName, sUser, sPassword
oClsErrorLog.ErrorLog "startMSDE-Connected . . ."

'Set the SQL Server path variable
HFACSMMain.gSQLServerPath =
osvr.Databases("master").PrimaryFilePath

startMSDE = True

ExitSub:
On Error GoTo 0
On Error Resume Next
osvr.DisConnect
Set osvr = Nothing
Screen.MousePointer = 0
Exit Function

StartError:
Screen.MousePointer = 0
If Err.Number = -2147023840 Then
'This error occurs when the server is already running,
'and Server.Start is executed on NT.
oClsErrorLog.ErrorLog "startMSDE-The server is
already started . . ."
Resume StartConnect
ElseIf Err.Number = -2147201024 Then
'This error occurs when server is already started and
connected.
oClsErrorLog.ErrorLog "startMSDE-The server is
already connected . . ."
Resume Next
ElseIf Err.Number = -2147023174 Then
'This error occurs when the server cannot be found.
oClsErrorLog.ErrorLog "startMSDE-The server could
not be found . . ."
MsgBox "Can't find server " & sSvrName & ". ",
vbCritical, _
"Connection Failed"
startMSDE = False
Resume ExitSub
ElseIf Err.Number = -2147203048 Then

If iSlowServerCounter < 2 Then
'Pause for 5 seconds while the server really restarts
HFACSMMain.gStrTextMessage = "A slow server was
detected. Giving extra time . . ."
HFACSMMain.gIntTimeToWait = 10
frmWait.Show 1
DoEvents 'Redraw screen
iSlowServerCounter = iSlowServerCounter + 1
Resume TimeoutResume:
End If

```

```

'This error occurs when the password or user ID is
wrong
oClsErrorLog.ErrorLog "startMSDE-Incorrect password
or user ID. . ."
MsgBox "Invalid User ID or Password.", vbCritical, _
"Connection Failed"
startMSDE = False
Resume ExitSub
ElseIf Err.Number = -2147221504 Then
'Logon timeout occurred
oClsErrorLog.ErrorLog "startMSDE-A harmless Login
timeout occurred . . ."
Resume Next
ElseIf Err.Number = -2147200991 Then
'This error occurs when replacing the database from a
file. It is
'caused because a current connection exists and NT
authentication is
'being attempted to stop the server.
oClsErrorLog.ErrorLog "startMSDE-A harmless lost
connection error occurred . . ."
Resume Next

'Or -2147023174

ElseIf Err.Number = -2147221163 Or -2147024891 Then
'This error occurs when attempting to log onto server
other than the
'local machine using SQLDMO.sqlserver. To work
around this switch
'to an ADO type connectio n.

oClsErrorLog.ErrorLog "SQLDMO connection failed.
Trying ADO . . ."

iADOAttemptCounter = iADOAttemptCounter + 1

'Try ADO 10 times, then give up
If iADOAttemptCounter = 10 Then
MsgBox "Tried 10 times"
GoTo FailedADO
End If

Dim oRemoteConnection As ADODB.Connection
Set oRemoteConnection = New ADODB.Connection
On Error GoTo FailedADO

If sNTAuth = "T" Then
oRemoteConnection.ConnectionString = _
"PROVIDER=SQLOLEDB.1;INTEGRATED
SECURITY=SSPI;" & _
"PERSIST SECURITY=FALSE;INITIAL
CATALOG=" & _
sDBName & ";DATA SOURCE=" & sSvrName
'& ";Network Library=dbmssocn"
Else
oRemoteConnection.ConnectionString = _
"PROVIDER=SQLOLEDB.1;PASSWORD=" _
& sPassword & ";PERSIST SECURITY
INFO=TRUE;USER ID=" & _
sUser & ";INITIAL CATALOG=" & sDBName _
& ";DATA SOURCE=" & sSvrName '&
";Network Library=dbmssocn"
End If

'Open the connection
oRemoteConnection.Open
oRemoteConnection.Close

```

```

'Destroy the connection since you verified it works
Set oRemoteConnection = Nothing

'If this connection exists, then the HFACS database
exists
'on the remote machine and no copy is needed, so set the
'global flag to true.
gblnNoCopyNeeded = True

startMSDE = True
Resume ExitSub
Else 'Unknown error
FailedADO:
oClsErrorLog.ErrorLog "Error: " & Err.Description & ",
Number: " & Err.Number
MsgBox "Destination host unreachable. The server may
not " & _
"be started or you may have to build a System DSN." &
Chr(13) & Chr(13) & _
"The detailed error message is: MSDE - " &
Err.Description & _
Chr(13) & Chr(13) & "Error Number: " & Err.Number,
-
vbOKOnly + vbCritical, "Connection Failed"
startMSDE = False
End If
Resume ExitSub

End Function

=====
'Function/Sub Name: copyMDF()
'
'Description: This procedure will check for the database on a
local
'Server. If the database does not exist, it will then copy and
install
'the HFACS database from the application path to the Server
data
'directory making a backup copy of the old database in case
an error
'occurs and a restore is needed.
'
'The last two copies of the database are kept in the server data
'directory in an attempt to prevent data loss.
'
'Input:
' bPerformCopy - Toggle to actually perform a copy or just
see if
' one is needed
' sSvrNameIn - The server to start
' sUserIn - The user ID with which to start the server
' sPasswordIn - The user password
' sMDFNameIn - The name of the MSDE Database to be
copied
' sDBNameIn - The name of the database
'
'Output: Success or Failure
'
'References:
' - Constructors.bas
' - HFACSMMain.bas
'
=====
'##ModelId=3B294D230242
Friend Function copyMDF(Optional bPerformCopy As
Boolean = True, Optional sSvrNameIn As String, Optional

```

```

sUserIn As String, Optional sPasswordIn As String, Optional
sMDFNameIn As String, Optional sDBNameIn As String)
As Boolean

```

```

Screen.MousePointer = 11
On Error GoTo StartError
Debug.Print
oClsErrorLog.ErrorLog "copyMDF-Copy routine initiated
... "
oClsErrorLog.ErrorLog "copyMDF-Creating new
SQLDMO object ... "

```

```

'Check for optional arguments and assign to defaults as
needed.

```

```

If Trim(sSvrNameIn) <> "" Then
sSvrName = sSvrNameIn
End If
If Trim(sUserIn) <> "" Then
sUser = sUserIn
End If
If Trim(sPasswordIn) <> "" Then
sPassword = sPasswordIn
End If
If Trim(sMDFNameIn) <> "" Then
sMDFName = sMDFNameIn
End If
If Trim(sDBNameIn) <> "" Then
sDBName = sDBNameIn
End If

```

```

'Declare an object for hard disk file manipulation
Dim FSO As Scripting.FileSystemObject

```

```

'Declare an object for SQL server manipulation
Dim osvr As sqlmo.SQLServer

```

```

'Declare a variable to hold the return value from the
'attachDB call
Dim strMessage As String

```

```

'For looping through databases on the server
Dim db As Database

```

```

'Declare an flag for determining if the database was
'found on the server
Dim fDataBaseFlag As Boolean

```

```

'The drive names used in FSO.Copyfile and
oSvr.AttachDB
'need to match the locations for Program Files and MSDE
on the
'user's machine.

```

```

'Initialize variables
Set FSO = CreateObject("Scripting.FileSystemObject")
Set osvr = CreateObject("sqlmo.sqlserver")
fDataBaseFlag = False

```

```

'Attempt a connection.
oClsErrorLog.ErrorLog "copyMDF-Attempting to
connect ... "
'If no login name, use NT Integrated security in an attempt
'to connect.
If sNTAuth = "T" Then
osvr.LoginSecure = True
End If

```

```

osvr.Connect sSvrName, sUser, sPassword

```

```

'Check for database on local MSDE Server
'by looping through all database names on the local MSDE
Server.
For Each db In osvr.Databases
    If db.Name = sDBName Then 'The database exists.
        oClsErrorLog.ErrorLog "copyMDF-The database
exists - " & _
            "No copy will be performed . . ."
        fDataBaseFlag = True
        copyMDF = True
        Exit For 'Get out of loop.
    End If
Next

If fDataBaseFlag = False Then 'There is no database name
match.

'Check to make sure the operation is being attempted on
'a local server.
If sSvrName <> "(local)" Then
    oClsErrorLog.ErrorLog "copyMDF-Failed check. "
& _
    "Can 't perform operation on remote server . . ."
    Screen.MousePointer = 0
    MsgBox "The server you are trying to connect to
exists," & _
        " but it is not the local machine or you have not " &
_
    "logged on as '(local)'. You be logged on to " & _
    "server '(local)' to perform this operation." & _
    Chr(13) & Chr(13) & "This program cannot create
a " & _
    "database on a machine other than the local
machine." & _
    vbCritical, "Connection Failed"
    Screen.MousePointer = 11
    copyMDF = False
    GoTo ExitSub
End If
oClsErrorLog.ErrorLog "copyMDF-Local machine is
the SQL server." & _
    " Continuing . . ."

'Copy file to data folder.
If bPerformCopy = True Then
    oClsErrorLog.ErrorLog "copyMDF-The HFACS
database was not found . . ."

    'We already ascertained that the db does not exist,
    'but the program can't overwrite the .mdf or _log.ldf
    'files if they exist, so if they exist, rename them
    'as BKP-1 and BKP-2, respectively. Permanently
    'delete any existing copy of a BKP-2 file.

    'Turn off error checking for the disk manipulation
    On Error GoTo 0
    On Error Resume Next
    FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _
    "BKP-1-" & sMDFName,
osvr.Databases("master").PrimaryFilePath & _
    "BKP-2-" & sMDFName, True
    FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _
    sMDFName,
osvr.Databases("master").PrimaryFilePath & _
    "BKP-1-" & sMDFName, True

```

```

FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _
    "BKP-1-" & Left(sMDFName, (Len(sMDFName) -
4)) & "_log.ldf", _
osvr.Databases("master").PrimaryFilePath & "BKP -
2-" & _
    Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", True
    FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _
    Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", _
osvr.Databases("master").PrimaryFilePath & "BKP -
1-" & _
    Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", True
    FSO.DeleteFile
osvr.Databases("master").PrimaryFilePath & _
    sMDFName, True
    FSO.DeleteFile
osvr.Databases("master").PrimaryFilePath & _
    Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", True

    'Now it's safe to copy the database from the
application
    'path to the server directory. Database updates (from
both
    'ftp and disks, are always first placed (copied or
downloaded)
    'into the application path – then they are copied to the
'server data directory by this function.
    FSO.CopyFile sInstDirectory & sMDFName, _
osvr.Databases("master").PrimaryFilePath &
sMDFName, True
    FSO.CopyFile sInstDirectory & Left(sMDFName,
(Len(sMDFName) - 4)) _
        & "_log.ldf",
osvr.Databases("master").PrimaryFilePath _
        & Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", True
    oClsErrorLog.ErrorLog "copyMDF-Function call
specified a copy was" & _
        " to be performed . . ."
    oClsErrorLog.ErrorLog "copyMDF-Copying " &
sDBName & " from " & _
        sInstDirectory & sMDFName & "to " & _
osvr.Databases("master").PrimaryFilePath & _
sMDFName & " . . ."

    'Attach the new .mdf file to the server.
    On Error GoTo 0
    On Error GoTo StartError
    strMessage = osvr.AttachDB(sDBName, "[" & _
osvr.Databases("master").PrimaryFilePath & _
sMDFName & "]")
    oClsErrorLog.ErrorLog "copyMDF-" & strMessage

    'This is a CRITICAL step that catches a failure to
attach
    'a new file.
    If Me.databaseExists = True Then
        copyMDF = True
    Else
        copyMDF = False
    End If

Else

```

```

        copyMDF = True
        oClsErrorLog.ErrorLog "copyMDF-Function call
specified not to copy" & _
        " the database. Ending . . ."
    End If
End If

ExitSub:
    Cleanup
    osvr.DisConnect
    oClsErrorLog.ErrorLog "copyMDF-Destroying the objects
created" & _
    " for copying hfacs.mdf . . ."
    Set osvr = Nothing
    Set FSO = Nothing
    Screen.MousePointer = 0
Exit Function

StartError:
    Screen.MousePointer = 0
    If Err.Number = -2147203048 Then
        "This error occurs when the password or user ID is
wrong
        oClsErrorLog.ErrorLog "copyMDF-Incorrect
password or user ID. . ."
        MsgBox "Invalid User ID or Password.", vbCritical, _
        "Connection Failed"
        copyMDF = False
        Resume ExitSub
    End If
    If Err.Number = -2147221504 Then
        "This error occurs when the user tries to connect a
'SQL 2K file to a SQL 7.0 compatible engine.
        oClsErrorLog.ErrorLog "copyMDF - This is a SQL
2K compatible file . . ."
        MsgBox "The file you are trying to attach is in SQL" &
_
        " 2000 format. The database engine on this machine is"
& _
        " configured for SQL 7.0.", vbCritical, "Connection
Failed"
        copyMDF = False
        Resume ExitSub
    End If
    'Unknown error
    oClsErrorLog.ErrorLog "Error: " & Err.Description & ",
Number: " & Err.Number
    MsgBox "Error copying database." & Chr(13) & Chr(13)
& _
    "The detailed error message is: " & Err.Description & _
    Chr(13) & Chr(13) & "Error Number: " & Err.Number, _
    vbOKOnly + vbCritical, "Copy Failed"
    copyMDF = False
    Resume ExitSub

End Function

'=====
Function/Sub Name: dropDB()
'
'Description: This procedure will check for the database on
the
'Server. If the database exists it will then permanently drop it.
'A normal drop specifies the bKillDBFiles paramater as
False, so
'A backup of the database is created before dropping it.

```

```

Passing a value of true for this parameter drops the database
with
'no backup.
'
'Input:
' bKillDBFiles - Toggle to drop the database without
backing-up
' sSvrNameIn - The server to start
' sUserIn - The user ID with which to start the server
' sPasswordIn - The user password
' sMDFNameIn - The name of the MSDE Database to be
copied
' sDBNameIn - The name of the database
'
'Output: Success or Failure
'
'References:
' - Constructors.bas
' - HFACSMMain.bas
'=====
##ModelId=3B294D2302FD
Friend Function dropDB(Optional bKillDBFiles As Boolean
= False, Optional sSvrNameIn As String, Optional sUserIn
As String, Optional sPasswordIn As String, Optional
sMDFNameIn As String, Optional sDBNameIn As String)
As Boolean

    Screen.MousePointer = 11
    On Error GoTo StartError
    Debug.Print
    oClsErrorLog.ErrorLog "dropDB-Drop routine was
initiated . . ."
    oClsErrorLog.ErrorLog "dropDB-Creating new
SQLDMO object . . ."

    'Check for optional arguments and assign to defaults as
needed.
    If Trim(sSvrNameIn) <> "" Then
        sSvrName = sSvrNameIn
    End If
    If Trim(sUserIn) <> "" Then
        sUser = sUserIn
    End If
    If Trim(sPasswordIn) <> "" Then
        sPassword = sPasswordIn
    End If
    If Trim(sMDFNameIn) <> "" Then
        sMDFName = sMDFNameIn
    End If
    If Trim(sDBNameIn) <> "" Then
        sDBName = sDBNameIn
    End If

    'Declare an object for hard disk file manipulation
    Dim FSO As Scripting.FileSystemObject

    'Declare an object for SQL server manipulation
    Dim osvr As sqlldmo.SQLServer

    'Declare a variable to hold the return value from the
'drop call
    Dim strMessage As String

    'For looping through databases on the server
    Dim db As Variant

    'Declare an flag for determining if the database
'was found on the server

```



```

Dim fDataBaseFlag As Boolean

'Initialize variables
Set FSO = CreateObject("Scripting.FileSystemObject")
Set osvr = CreateObject("sqlldmo.sqlserver")
fDataBaseFlag = False

'Attempt a connection.
oClsErrorLog.ErrorLog "dropDB-Attempting to connect . . ."
'If no login name, use NT Integrated security in an attempt.
'to connect.
If sNTAuth = "T" Then
    osvr.LoginSecure = True
End If
osvr.Connect sSvrName, sUser, sPassword

'Check to make sure the operation is being attempted on
'a local server.
If sSvrName <> "(local)" Then
    dropDB = False
    oClsErrorLog.ErrorLog "dropDB-Failed check. Can't
perform " & _
        "operation on remote server . . ."
    Screen.MousePointer = 0
    MsgBox "The server you are trying to connect to
exists," & _
        " but it is not the local machine or you have not
logged" & _
        " on as '(local)'. You be logged on to server '(local)'"
    & _
        " to perform this operation." & Chr(13) & Chr(13) & _
        "This program cannot create a database on a machine
other" & _
        " than the local machine.", vbCritical + vbOKOnly, _
        "Connection Failed"
    Screen.MousePointer = 11
    GoTo ExitSub
End If

oClsErrorLog.ErrorLog "dropDB-Local machine is the
SQL server. Continuing . . ."

'Check for database on local MSDE Server
'by looping through all database names on the local MSDE
Server.
For Each db In osvr.Databases
    If db.Name = sDBName Then 'The database exists.
        oClsErrorLog.ErrorLog "dropDB-The database exists
on the server . . ."
        fDataBaseFlag = True
        Exit For 'Get out of loop.
    End If
Next

If fDataBaseFlag = True Then 'There is a database name
match.
    'drop the database.
    oClsErrorLog.ErrorLog "dropDB-The HFACS database
was found . . ."
    oClsErrorLog.ErrorLog "dropDB-Dropping " &
sDBName & " from " & _
        sSvrName & " . . ."
    strMessage = osvr.DetachDB(sDBName, True)

    'Print any error messages from the server

oClsErrorLog.ErrorLog "dropDB-" & strMessage

'NOTE: Uncomment this to see the SQL drop server
messages
'MsgBox strMessage

'Check to make sure the drop was successful.
If Me.databaseExists = True Then
    Screen.MousePointer = vbDefault
    MsgBox "There was an error dropping the existing"
& _
    " database file from the database." & Chr(13) & _
    Chr(13) & "The new file will not be installed.", _
    vbExclamation + vbOKOnly, "Database Drop
Failed"
    dropDB = False
    GoTo ExitSub
End If

'Turn off error checking for the disk manipulation
On Error GoTo 0
On Error Resume Next
If bKillDBFiles = True Then
    'The user specified to physically delete the .mdf files
    'from the server with no backup.
    oClsErrorLog.ErrorLog "dropDB-Deletion of files
was requested as" & _
        " well. Deleting files . . ."
    FSO.DeleteFile
osvr.Databases("master").PrimaryFilePath & _
        sMDFName, True
    FSO.DeleteFile
osvr.Databases("master").PrimaryFilePath & _
        Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", True
    dropDB = True
Else
    *** NOTE: This functionality is turned off because
the copy
    'routine accomplishes the backing of files, but this
code is
    'left here for reuse purposes. Just uncomment to use
this
    'functionality.

    'Otherwise rename the old ones
    'FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _
        "BKP-1-" & sMDFName,
osvr.Databases("master").PrimaryFilePath & _
        "& "BKP-2-" & sMDFName, True
    'FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _
        sMDFName,
osvr.Databases("master").PrimaryFilePath & _
        "BKP-1-" & sMDFName, True
    'FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _
        "BKP-1-" & Left(sMDFName, (Len(sMDFName)
- 4)) & _
        "_log.ldf",
osvr.Databases("master").PrimaryFilePath & _
        "BKP-2-" & Left(sMDFName, (Len(sMDFName)
- 4)) & _
        "_log.ldf", True
    'FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _

```

```

        'Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", _
osvr.Databases("master").PrimaryFilePath &
"BKP-1-" & _
'Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", True
'FSO.DeleteFile
osvr.Databases("master").PrimaryFilePath & _
'sMDFName, True
'FSO.DeleteFile
osvr.Databases("master").PrimaryFilePath & _
'Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", True
dropDB = True
End If
On Error GoTo 0
On Error GoTo StartError

Else

oClsErrorLog.ErrorLog "dropDB-The HFACS database
was not found, so " & _
"no drop is necessary . . ."
Turn off error checking for the disk manipulation
On Error GoTo 0
On Error Resume Next
If bKillDBFiles = True Then
'Physically delete the .mdf files from the server
oClsErrorLog.ErrorLog "dropDB-Deletion of files
was requested, however." & _
"Deleting files . . ."
FSO.DeleteFile
osvr.Databases("master").PrimaryFilePath & _
sMDFName, True
FSO.DeleteFile
osvr.Databases("master").PrimaryFilePath & _
'Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", True
Else
*** NOTE: This functionality is turned off because
the copy routine
'accomplishes the backing of files, but this code is left
here for
'reuse purposes. Just uncomment to use this
functionality.

'Otherwise rename the old ones
'FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _
"BKP-1-" & sMDFName,
osvr.Databases("master").PrimaryFilePath & _
'& "BKP-2-" & sMDFName, True
'FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _
'sMDFName,
osvr.Databases("master").PrimaryFilePath & _
"BKP-1-" & sMDFName, True
'FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _
"BKP-1-" & Left(sMDFName, (Len(sMDFName) -
4)) & _
'_log.ldf",
osvr.Databases("master").PrimaryFilePath & _
"BKP-2-" & Left(sMDFName, (Len(sMDFName) -
4)) & _
'_log.ldf", True
'FSO.CopyFile
osvr.Databases("master").PrimaryFilePath & _

```

```

        'Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", _
osvr.Databases("master").PrimaryFilePath &
"BKP-1-" & _
'Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", True
'FSO.DeleteFile
osvr.Databases("master").PrimaryFilePath & _
'sMDFName, True
'FSO.DeleteFile
osvr.Databases("master").PrimaryFilePath & _
'Left(sMDFName, (Len(sMDFName) - 4)) &
"_log.ldf", True
End If
On Error GoTo 0
On Error GoTo StartError
dropDB = True
End If

ExitSub:
osvr.DisConnect
oClsErrorLog.ErrorLog "dropDB-Destroying the objects
created for dropping" & _
"the database . . ."
Set osvr = Nothing
Set FSO = Nothing
Screen.MousePointer = 0
Exit Function

StartError:
Screen.MousePointer = 0
If Err.Number = -2147203048 Then
'This error occurs when the password or user ID is
wrong
oClsErrorLog.ErrorLog "dropDB-Incorrect password
or user ID. . ."
MsgBox "Invalid User ID or Password.", vbCritical, _
"Connection Failed"
dropDB = False
Resume ExitSub
End If
'Unknown Error
oClsErrorLog.ErrorLog "Error: " & Err.Description & ",
Number: " & Err.Number
MsgBox "Error dropping database." & Chr(13) & Chr(13)
& _
"The detailed error message is: " & Err.Description &
Chr(13) _
& Chr(13) & "Error Number: " & Err.Number, _
vbOKOnly + vbCritical, "Database Drop Failed"
dropDB = False
Resume ExitSub

End Function

```

```

'=====
Function/Sub Name: databaseExists()
'
'Description: This procedure will connect to a SQL server
that is
'already running and determine if a database exists.
'
'Input:
' sSvrNameIn - The server to start
' sUserIn - The user ID with which to start the server
' sPasswordIn - The user password

```

```

' sMDFNameIn - The name of the MSDE Database to be
copied
' sDBNameIn - The name of the database
'
'Output: Success or Failure
'
'References:
' - Constructors.bas
' - HFACSMMain.bas
'=====
'##ModelId=3B294D2303A9
Friend Function databaseExists(Optional sSvrNameIn As
String, Optional sUserIn As String, Optional sPasswordIn As
String, Optional sMDFNameIn As String, Optional
sDBNameIn As String) As Boolean

    Screen.MousePointer = 11
    On Error GoTo StartError
    Debug.Print
    oClsErrorLog.ErrorLog "databaseExists-Connect
routine was initiated . . ."
    oClsErrorLog.ErrorLog "databaseExists-Creating new
SQLDMO object . . ."

    'Check for optional arguments and assign to defaults as
needed.
    If Trim(sSvrNameIn) <> "" Then
        sSvrName = sSvrNameIn
    End If
    If Trim(sUserIn) <> "" Then
        sUser = sUserIn
    End If
    If Trim(sPasswordIn) <> "" Then
        sPassword = sPasswordIn
    End If
    If Trim(sMDFNameIn) <> "" Then
        sMDFName = sMDFNameIn
    End If
    If Trim(sDBNameIn) <> "" Then
        sDBName = sDBNameIn
    End If

    'Declare an object for SQL server manipulation
    Dim osvr As sqlldmo.SQLServer

    'For looping through databases on the server
    Dim db As Variant

    'Declare an flag for determining if the database was found
on
    'the server
    Dim fDataBaseFlag As Boolean

    'Initialize variables
    Set osvr = CreateObject("SQLDMO.sqlserver")
    fDataBaseFlag = False

    'Attempt a connection.
    oClsErrorLog.ErrorLog "databaseExists-Attempting to
connect . . ."
    'If no login name, use NT Integrated security in an attempt.
to connect.
    If sNTAuth = "T" Then
        osvr.LoginSecure = True
    End If
    osvr.Connect sSvrName, sUser, sPassword

    'Check for database on Server

```

'by looping through all database names on the local MSDE
Server.

```

    For Each db In osvr.Databases
        If db.Name = sDBName Then 'The database exists.
            oClsErrorLog.ErrorLog "databaseExists-The
database exists . . ."
            fDataBaseFlag = True
            databaseExists = True
            Exit For 'Get out of loop.
        End If
    Next

    If fDataBaseFlag = False Then 'There is no database name
match.
        databaseExists = False
    End If

ExitSub:
    'Turn off error checking so that errors in destroying objects
'don't cause an endless loop.
    On Error GoTo 0
    On Error Resume Next
    osvr.DisConnect
    oClsErrorLog.ErrorLog "databaseExists-Destroying the
objects created for" & _
        " checking database existence . . ."
    Set osvr = Nothing
    Screen.MousePointer = 0
Exit Function

StartError:
    Screen.MousePointer = 0
    If Err.Number = -2147203048 Then
        'This error occurs when the password or user ID is
wrong
        oClsErrorLog.ErrorLog "databaseExists-Incorrect
password or user ID . . ."
        MsgBox "Invalid User ID or Password.", vbCritical, _
            "Connection Failed"
        databaseExists = False
        Resume ExitSub
    Else
        'Unknown error. Don't show any message box
        oClsErrorLog.ErrorLog "Error: " & Err.Description & ",
Number: " & Err.Number
        databaseExists = False
        Resume ExitSub
    End If

```

End Function

```

'=====
'Function/Sub Name: StartAndCopy()
'
'Description: This procedure combines the functionality of
the
'startMSDE() and copyMDF() functions with the added
ability to
'determine if a copy is needed based upon the results of the
'startMSDE() call. For example, if a remote connection is
attempted
'and succeeds, startMSDE() will return True, but no copy will
be
'neccessary.
'
'In addition, this function detects if a copy failed and will
'attempt to repair the database by offering an option to restore

```

'an old copy of the database. This is useful when called from  
'a failed FTP update attempt.

'Input:

' sSvrNameIn - The server to start  
' sUserIn - The user ID with which to start the server  
' sPasswordIn - The user password  
' sMDFNameIn - The name of the MSDE Database to be  
copied  
' sDBNameIn - The name of the database

'Output: Success or Failure

'References:

' - Constructors.bas  
' - HFACSMMain.bas

=====

##ModelId=3B294D24005D  
Friend Function StartAndCopy(Optional sSvrNameIn As  
String, Optional sUserIn As String, Optional sPasswordIn As  
String, Optional sMDFNameIn As String, Optional  
sDBNameIn As String) As Boolean

'Check for optional arguments and assign to defaults as  
needed.

If Trim(sSvrNameIn) <> "" Then  
    sSvrName = sSvrNameIn  
End If  
If Trim(sUserIn) <> "" Then  
    sUser = sUserIn  
End If  
If Trim(sPasswordIn) <> "" Then  
    sPassword = sPasswordIn  
End If  
If Trim(sMDFNameIn) <> "" Then  
    sMDFName = sMDFNameIn  
End If  
If Trim(sDBNameIn) <> "" Then  
    sDBName = sDBNameIn  
End If

'Test result variables

Dim bTestSuccess1 As Boolean  
Dim bTestSuccess2 As Boolean

bTestSuccess1 = Me.startMSDE()

'If logging to a remote machine, the startMSDE will verify  
'that the database exists and set this flag. No copy will  
'be needed, because the database exists on the remote  
server.

If gblnNoCopyNeeded = True Then StartAndCopy = \_  
    True: Exit Function

'Only copy if the start was a success

If bTestSuccess1 = True Then  
    bTestSuccess2 = Me.copyMDF()  
End If

DoEvents 'Redraw the screen

'if the copy failed, attempt restore of old DB.

If bTestSuccess2 = False Then  
    oClsErrorLog.ErrorLog "StartAndCopy-Trying to restore  
the old DB . . ."

'Don't try to restore if this is the first time DB has  
'been run

If HFACSMMain.gStrFirstRun = "T" Then  
    StartAndCopy = False  
    Exit Function  
End If

If Me.restoreOldDB = False Then  
    StartAndCopy = False  
    Exit Function

Else  
    StartAndCopy = True  
    Exit Function  
End If

Else  
    StartAndCopy = True  
End If

End Function

=====

'Function/Sub Name: restoreOldDB()  
'

'Description: This function is called when a copy operation  
fails and  
'there is no HFACS database file attached to the local server.  
Once  
'called, this function prompts the user to restore the old  
database.

'If the user opts to restore the database, a restore is first  
attempted  
'using the current logon information. If this attempt fails, a  
second  
'attempt is made as a "last -ditch" effort using the "sa" logon  
and  
'no password. If both attempts fail, the database will not be  
installed  
'on the local server and the HFACS program will not  
function. System

'Administrator assistance will be required to attach a copy of  
the  
'database.

'Input:

' sSvrNameIn - The server to start  
' sUserIn - The user ID with which to start the server  
' sPasswordIn - The user password  
' sMDFNameIn - The name of the MSDE Database to be  
copied  
' sDBNameIn - The name of the database

'Output: Success or Failure

'References:

' - Constructors.bas  
' - HFACSMMain.bas

=====

##ModelId=3B294D2400FA  
Friend Function restoreOldDB(Optional sSvrNameIn As  
String, Optional sUserIn As String, Optional sPasswordIn As  
String, Optional sMDFNameIn As String, Optional  
sDBNameIn As String) As Boolean

On Error GoTo StartError

'Check for optional arguments and assign to defaults as  
needed.

```

If Trim(sSvrNameIn) <> "" Then
    sSvrName = sSvrNameIn
End If
If Trim(sUserIn) <> "" Then
    sUser = sUserIn
End If
If Trim(sPasswordIn) <> "" Then
    sPassword = sPasswordIn
End If
If Trim(sMDFNameIn) <> "" Then
    sMDFName = sMDFNameIn
End If
If Trim(sDBNameIn) <> "" Then
    sDBName = sDBNameIn
End If

Dim response As Variant
response = MsgBox("HFACS was unable to install a new
update" & _
    " or something is preventing it from finding the
database" & _
    " on the local machine." & Chr(13) & Chr(13) & _
    "If you recieved this message after trying to perform an"
& _
    " update via disk or FTP, then you should revert to the"
& _
    " previous copy of the database." & Chr(13) & Chr(13)
& _
    "Do you want to revert to the previous copy of the
database?", _
vbYesNo + vbDefaultButton1 + vbExclamation, _
    "Problem Finding Database")

If response = vbYes Then 'Attempt to restore the old DB

'Declare an object for hard disk file manipulation
Dim FSO As Scripting.FileSystemObject
Set FSO = CreateObject("Scripting.FileSystemObject")

DoEvents 'Redraw the screen

'Atmpt to revert to the old copy of the database
Screen.MousePointer = vbHourglass

'Turn off error checking for disk manipulation
On Error GoTo 0
On Error Resume Next
FSO.DeleteFile HFACSMMain.gStrAppPath & _
    HFACSMMain.gStrDatabaseFileName, True
FSO.CopyFile HFACSMMain.gStrAppPath &
sMDFName & _
    ".old", HFACSMMain.gStrAppPath & sMDFName,
True
FSO.DeleteFile HFACSMMain.gStrAppPath & _
    Left(HFACSMMain.gStrDatabaseFileName, _
        (Len(sMDFName) - 4)) & "_log.ldf", True
FSO.CopyFile HFACSMMain.gStrAppPath & _
    Left(HFACSMMain.gStrDatabaseFileName, _
        (Len(HFACSMMain.gStrDatabaseFileName) - 4)) & _
    "_log.ldf.old", HFACSMMain.gStrAppPath & _
    Left(HFACSMMain.gStrDatabaseFileName, _
        (Len(HFACSMMain.gStrDatabaseFileName) - 4)) & _
    "_log.ldf", True
On Error GoTo 0
On Error GoTo StartError

'Now try to copy it to the Server
Dim bTestSuccess As Boolean

```

```

bTestSuccess = Me.copyMDF
If bTestSuccess = False Then

    'If that didn't work, then revert to the original
    'system settings and try one last time.
    HFACSMMain.gStrUID = "sa"
    HFACSMMain.gStrPWD = ""
    HFACSMMain.gStrServerName = "(local)"
    HFACSMMain.gStrDatabaseFileName =
"HFACS.mdf"
    HFACSMMain.gStrDatabaseName = "HFACS"
    HFACSMMain.gStrAppPath =
HFACSMMain.gStrAppPath
    HFACSMMain.gStrAutoLogon = "F"
    HFACSMMain.gStrFirstRun = "F"
    HFACSMMain.gStrNTauth = "F"
    HFACSMMain.gStrTypeDB =
HFACSMMain.gStrTypeDB

    Dim bLastTryWDefaultSettings As Boolean
    bLastTryWDefaultSettings = Me.copyMDF(True,
"(local)", _
    "sa", , "HFACS.mdf", "HFACS")

    'If that failed inform the user of the problem.
    If bLastTryWDefaultSettings = False Then
        Screen.MousePointer = vbDefault
        MsgBox "A fatal error has ocured and HFACS
has " & _
            "become corrupted." & Chr(13) & Chr(13) & _
            "Please contact your system administrator to " & _
            "replace the corrupted files.", vbOKOnly, _
            "Fatal Error - HFACS Is Corrupted"
        restoreOldDB = False
        GoTo ExitSub
    Else
        restoreOldDB = True
        GoTo ExitSub
    End If

Else
    restoreOldDB = True
End If

Else
    'Just exit.
    restoreOldDB = False
    Exit Function

End If

ExitSub:
    DoEvents 'Redraw the screen
    Screen.MousePointer = vbDefault
    Set FSO = Nothing

Exit Function

StartError:
    Screen.MousePointer = 0
    'Unknown error
    oClsErrorLog.ErrorLog "Error: " & Err.Description & ",
Number: " & Err.Number
    MsgBox "An error occurred restoring the database." &
Chr(13) & _
        Chr(13) & "The detailed error message is: " & _
        Err.Description & Chr(13) & Chr(13) & _

```

```

        "Error Number: " & Err.Number, _
        vbOKOnly + vbCritical, "Error Restoring Database"
    restoreOldDB = False
    Resume ExitSub
End Function

'=====
'Function/Sub Name: restartMSDE()
'
'Description: Before an .mdf database file can be dropped
and a new
'file attached, all users must be logged off. This function
stops and
'restarts the server effectively ensuring all users are logged
off
'and that the server services are refreshed. This function can
only
'be used in conjunction with an update operation (either disk
or FTP)
'as it also copies the file from the download/temp copy
directory
'(which is the application path) to the server data directory.
This
'copy can only be performed when the server is stopped.
'
'Input:
' sSvrNameIn - The server to start
' sUserIn - The user ID with which to start the server
' sPasswordIn - The user password
'
'Output: Success or Failure
'
'References:
' - Constructors.bas
' - HFACSMMain.bas
' - frmWait.frm
'=====
##ModelId=3B294D240196
Friend Function restartMSDE(Optional sSvrNameIn As
String, Optional sUserIn As String, Optional sPasswordIn As
String) As Boolean

    Screen.MousePointer = 11
    On Error GoTo StartError

    'Check for optional arguments and assign to defaults as
needed.
    If Trim(sSvrNameIn) <> "" Then
        sSvrName = sSvrNameIn
    End If
    If Trim(sUserIn) <> "" Then
        sUser = sUserIn
    End If
    If Trim(sPasswordIn) <> "" Then
        sPassword = sPasswordIn
    End If

    *****
    'Remove this to allow restarting of other servers than local.
    sSvrName = "(local)"

    'Declare an object for SQL server manipulation
    Dim osvr As sqlDMO.SQLServer

    'Create the SQLDMO Server Object.

    Set osvr = CreateObject("SQLDMO.sqlserver")

    osvr.LoginTimeout = 20
    'Start Server.

    'The server must be started and connected in order to stop
it.
    'Attempt a connection.
    'If no login name, use NT Integrated security in an attempt
'to connect.
    If sNTAuth = "T" Then
        osvr.LoginSecure = True
    End If

    'This is the actual connection attempt.
    osvr.Connect sSvrName, sUser, sPassword

    'Create a temp variable for the path to the server DB files
'because once the server is stopped, you can't access the
osvr object
    Dim sPathToServer As String
    sPathToServer =
osvr.Databases("master").PrimaryFilePath

    On Error GoTo 0
    'The shutdown command causes an error because the
current
'connection is lost, so resume next.
    On Error Resume Next
    oClsErrorLog.ErrorLog "restartMSDE-Attempting to stop
server. . ."
    osvr.Shutdown (True)
    oClsErrorLog.ErrorLog "restartMSDE-The server was
successfully stopped . . ."
    Set osvr = Nothing

    HFACSMMain.gStrTextMessage = "Stopping the server . . ."
    HFACSMMain.gIntTimeToWait = 15

    'This keeps the form visible when it loses the focus
    Screen.ActiveForm.AutoRedraw = False

    'Pause for 15 seconds while the server really restarts
    frmWait.Show 1
    Screen.ActiveForm.AutoRedraw = True

    'Repaint the frmFtpUpdate form if it's open
    oClsErrorLog.ErrorLog "Is FTP form open? => " & _
HFACSMMain.IsOpen("frmFtpUpdate")
    If HFACSMMain.IsOpen("frmFtpUpdate") Then
        frmFtpUpdate.Refresh
        oClsErrorLog.ErrorLog "restartMSDE-Attempting to
restart server . . ."
        DoEvents 'Redraw screen

        'Repaint the frmFtpUpdate form if it's open
        oClsErrorLog.ErrorLog "Is frmDiskUpdate form open? =>
" & _
HFACSMMain.IsOpen("frmDiskUpdate")
        If HFACSMMain.IsOpen("frmDiskUpdate") Then
            frmDiskUpdate.Refresh
            oClsErrorLog.ErrorLog "restartMSDE-Attempting to
restart server . . ."
            DoEvents 'Redraw screen
        End If
    End If

    *****

```

```

'This block is responsible for copying the current db file to
'the local AppPath as the most current backup (.old) this is
'the file that will be restored in the event of catastrophic
'failure. It can only be accomplished here because the
SQL
'server has to be stopped.

```

```

*****

```

```

'Turn off error checking for disk manipulation
On Error GoTo 0
On Error Resume Next

'Declare an object for hard disk file manipulation
Dim FSO As Scripting.FileSystemObject
Set FSO = CreateObject("Scripting.FileSystemObject")

'Copy the last backup to the AppPath as the last good
backup.
oClsErrorLog.ErrorLog
*****
oClsErrorLog.ErrorLog "Copying the most recent files
to the AppPath "
oClsErrorLog.ErrorLog "while the server is stopped."
oClsErrorLog.ErrorLog
*****
FSO.CopyFile sPathToServer &
HFACSMMain.gStrDatabaseFileName, _
HFACSMMain.gStrAppPath &
HFACSMMain.gStrDatabaseFileName & _
".old", True

FSO.CopyFile sPathToServer &
Left(HFACSMMain.gStrDatabaseFileName, _
(Len(HFACSMMain.gStrDatabaseFileName) - 4)) &
"_log.ldf", _
HFACSMMain.gStrAppPath &
Left(HFACSMMain.gStrDatabaseFileName, _
(Len(HFACSMMain.gStrDatabaseFileName) - 4)) & _
"_log.ldf.old", True

Set FSO = Nothing

```

```

'Turn on error checking
On Error GoTo 0
On Error GoTo StartError

```

```

'Start the server back up again.
Me.startMSDE

```

```

'Pause for 5 seconds while the server really restarts
HFACSMMain.gStrTextMessage = "Starting the server . . ."
HFACSMMain.gIntTimeToWait = 5
frmWait.Show 1
DoEvents 'Redraw screen
On Error GoTo 0

```

```

restartMSDE = True

```

```

ExitSub:
Screen.MousePointer = 0
Exit Function

```

```

StartError:
Screen.MousePointer = 0
oClsErrorLog.ErrorLog "Error: " & Err.Description & ",
Number: " & Err.Number
MsgBox "An error occurred restarting the server." &
Chr(13) & _
& Chr(13) & "The detailed error message is: " & _
Err.Description & Chr(13) & Chr(13) & _
"Error Number: " & Err.Number, _
vbOKOnly + vbCritical, "Problem Restarting Server"
restartMSDE = False
Resume ExitSub

```

```

End Function

```

```

Private Sub Class_Terminate()

```

```

Set oClsErrorLog = Nothing

```

```

End Sub

```

## **CLASS-MUpdateController**

Option Explicit

```
#####
CLASS DESCRIPTION
#####
'Class Name: UpdateController.cls
'
'Author: Pat Flanders & Scott Tufts
'
'This class is the controller class for the cFTP class, the FTP
'form (frmFTPUpdate), and the common dialog control for
'reading an update
'from a disk.
'
'References:
' - Microsoft Windows Common Controls 6.0
'
' NOTE: See function headers for internal component
'references.
#####
```

```
*****
FUNCTIONS
*****
```

```
=====
Function/Sub Name: getUpdate()
'
'Description: 'This function initiates the FTP update session
'by
'creating an instance of frmFtpUpdate which actually
'performs the
'download and update.
'
'Input: None
'
'Output: Success or Failure
'
'References:
' - frmFtpUpdate.frm
'=====
##ModelId=3B294D0D03C8
Friend Function getUpdate() As Boolean

    frmFtpUpdate.Show 1
    'Return results of the FTP session
    If gblnFTPSuccess = True Then getUpdate = True Else
    getUpdate = False

End Function
```

```
=====
Function/Sub Name: getUpdateDisk()
'
'Description: This function displays the "Open" dialog box
'from
'the Microsoft Windows Common Controls 6.0 allowing the
'user to
```

```
'identify a path on a disk/network share where the
HFACS.mdf/_log.lsf
'update files reside. It then copies the files to the application
'path on the local machine and instantiates an instance of
'frmDiskUpdate to install them.
'
```

```
'Input: None
'
'Output: Success or Failure
'
'References:
' - frmDiskUpdate.frm
'=====
```

```
##ModelId=3B294D0E000F
Friend Function getUpdateDisk() As Boolean
```

On Error GoTo StartError

```
'Check to make sure user is updating the local server
If HFACSMMain.gStrServerName <> "(local)" Then
    MsgBox "You can only perform an update when logged
into " & _
        "the '(local)' server.", _
        vbExclamation + vbOKOnly, "Can't Update"
    getUpdateDisk = False
    GoTo ExitSub
End If
```

```
'Create a dialog box object
Dim oDialog As New MSComDlg.CommonDialog
```

```
'Variable to hold the path and file to get.
Dim sFileName As String
```

```
' Set CancelError is True
oDialog.CancelError = True
' Set flags
oDialog.Flags = cdIOFNHideReadOnly
' Set filters
oDialog.Filter = "HFACS Database Files
(HFACS.mdf)|HFACS.mdf"
' Specify default filter
oDialog.FilterIndex = 1
' Display the Open dialog box
oDialog.ShowOpen
sFileName = oDialog.FileName
```

DoEvents 'Redraw the screen

```
Dim bDiskUpdateSuccess As Boolean
bDiskUpdateSuccess =
frmDiskUpdate.performDiskUpdate(sFileName)
```

```
If bDiskUpdateSuccess = True Then
    MsgBox "The HFACS update was successfully
installed!" & _
        Chr(13) & Chr(13) & "HFACS will now re-
initialize.", _
        vbInformation + vbOKOnly, "Finished"
    getUpdateDisk = True
Else
    getUpdateDisk = False
End If
```



```
ExitSub:
    Set oDialog = Nothing
Exit Function
StartError:
```

```
Screen.MousePointer = vbDefault
getUpdateDisk = False
Resume ExitSub
End Function
```

## **FORMCLASS-frmDiskUpdate**

Option Explicit

```
#####
' FORM DESCRIPTION
#####
'Class Name: frmDiskUpdate.frm
'
'Author: Pat Flanders & Scott Tufts
'
'This class is responsible for performing an update of the
HFACS
'database from a disk/network share.
'
'References: None
'
' NOTE: See function headers for internal component
references.
#####
```

```
*****
' FUNCTIONS
*****
```

```
=====
'Function/Sub Name: performDiskUpdate()
'
'Description: This function performs the actual update,
updating
'the form as it progresses.
'
'Input:
' sFileToGet - Path to the HFACS.mdf and HFACS_log.ldf
files
'
' used to update the database.
'
'Output: Success or Failure
'
'References:
' - Constructors.bas
' - MSDE.cls
' - HFACSMMain.bas
'
=====
```

```
##ModelId=3B294D160242
Friend Function performDiskUpdate(sFileToGet As String)
As Boolean
```

Me.Visible = True

On Error GoTo vbErrorHandler

Screen.MousePointer = vbHourglass

```
'Update the form
frmDiskUpdate.lblAction.Caption = "Getting the new file .
.."
frmDiskUpdate.lblAction.Refresh
```

```
'Declare an object for hard disk file manipulation
Dim FSO As Scripting.FileSystemObject
Set FSO = CreateObject("Scripting.FileSystemObject")
```

```
'Turn off error checking for disk manipulation
On Error GoTo 0
On Error Resume Next
'Copy the file . . . renaming it to HFACS.mdf
FSO.DeleteFile HFACSMMain.gStrAppPath & _
HFACSMMain.gStrDatabaseFileName, True
FSO.CopyFile sFileToGet, HFACSMMain.gStrAppPath & _
HFACSMMain.gStrDatabaseFileName, True
FSO.DeleteFile HFACSMMain.gStrAppPath & _
Left(HFACSMMain.gStrDatabaseFileName, _
(Len(HFACSMMain.gStrDatabaseFileName) - 4)) &
"_log.ldf", True
Dim sTempJustTheFileName As String
Dim sTempJustThePathPart As String
sTempJustTheFileName = Right(sFileToGet, _
Len(HFACSMMain.gStrDatabaseFileName))
sTempJustThePathPart = Left(sFileToGet, _
(Len(sFileToGet) -
Len(HFACSMMain.gStrDatabaseFileName)))
FSO.CopyFile sTempJustThePathPart & _
Left(sTempJustTheFileName, _
(Len(sTempJustTheFileName) - 4)) & _
"_log.ldf", HFACSMMain.gStrAppPath &
Left(sTempJustTheFileName, _
(Len(sTempJustTheFileName) - 4)) & "_log.ldf", True
On Error GoTo 0
On Error GoTo vbErrorHandler
```

```
'Update the form
frmDiskUpdate.lblAction.Caption = "Installing . . ."
frmDiskUpdate.lblAction.Refresh
```

```
'Install the new File
Dim bTestSuccess As Boolean
bTestSuccess = True
bTestSuccess =
Constructors.New_MSDE(HFACSMMain.gStrUID, _
HFACSMMain.gStrPWD, _
HFACSMMain.gStrServerName, _
HFACSMMain.gStrDatabaseFileName, _
HFACSMMain.gStrDatabaseName, _
HFACSMMain.gStrAppPath, _
HFACSMMain.gStrAutoLogon, _
HFACSMMain.gStrFirstRun, _
HFACSMMain.gStrNTauth, _
HFACSMMain.gStrTypeDB)
```

```
'Update the form
frmDiskUpdate.lblAction.Caption = _
"Stopping and restarting server . . ."
frmDiskUpdate.lblAction.Refresh
```

```
'Restart MSDE
oMSDE.restartMSDE
```

```
'Update the form
frmDiskUpdate.lblAction.Caption = "Dropping old
database . . ."
frmDiskUpdate.lblAction.Refresh
```

```
'Drop the old file
If oMSDE.dropDB <> True Then
performDiskUpdate = False
```

```

        GoTo ExitSub
    End If

    'Update the form
    frmDiskUpdate.lblAction.Caption = "Attaching new file . .
."
    frmDiskUpdate.lblAction.Refresh

    'Start and copy the new file over
    If oMSDE.StartAndCopy <> True Then
        performDiskUpdate = False
        GoTo ExitSub
    End If

    Screen.MousePointer = vbDefault

    performDiskUpdate = True

ExitSub:
    'Cleanup
    Set oMSDE = Nothing

```

```

        Set FSO = Nothing
        Me.Visible = False

Exit Function

vbErrorHandler:
    frmDiskUpdate.lblAction.Caption = Err.Description
    frmDiskUpdate.lblAction.Refresh
    MsgBox "An error occurred trying to install the files.
Verify" & _
    " that you have adequate permissions to perform this
update." & _
    Chr(13) & Chr(13) & "The detailed error message is: " &
    _
    Err.Description & Chr(13) & Chr(13) & "Error Number: "
    & Err.Number, _
    vbOKOnly + vbCritical, "Error During Install"
    performDiskUpdate = False
    Resume ExitSub

End Function

```

## **FORMCLASS-frmFtpUpdate**

Option Explicit

```
'#####  
' FORM DESCRIPTION  
'#####  
'Class Name: frmFtpUpdate.frm  
'  
'Author: Pat Flanders & Scott Tufts  
'  
'This class is responsible for performing an update of the  
HFACS  
'database via FTP. This class uses the FTPServer.exe server  
and  
'the CallbackCls.cls to receive status messages from the  
HFACS  
'FTP server.  
'  
'ASIDE: The FTP server (HFACSFTP.exe) provides the  
functions needed  
'to get FTP updates. These functions and their associated  
classes  
'were removed from this component and compiled separately  
in order  
'to work around the inability of Visual Basic to provide  
support  
'for free threading. By placing the FTP functionality in a  
'separately compiled executable, it can run in it's own  
process,  
'which allows screen updates during long FTP downloads.  
'  
'References:  
' - Microsoft Data Formating Object Library 6.0  
' - Microsoft Scripting Runtime  
' - GIF89 1.0 (For animated GIFs on Forms)  
' - The HFACSFTP.exe ftp server.  
'  
'NOTE: See function headers for internal component  
references.  
'#####
```

```
*****  
' PROPERTIES  
'*****
```

```
'Object variable for holding an instance of the FTPserver  
##ModelId=3B294D1F032D  
Dim oDoFTPThread As HFACSFTP.cFTP
```

```
'A temp string variable to simplify string manipulation when  
'determining paths on the FTP server and for download  
locations  
##ModelId=3B294D1F0399  
Dim sTempJustTheFileName As String
```

```
*****  
' FUNCTIONS  
'*****
```

```
'=====
```

```
'Function/Sub Name: cmdCancel_Click()
```

```
'Description: This sub closes the form
```

```
'Input: None
```

```
'Output: None
```

```
'References: None
```

```
'=====
```

```
##ModelId=3B294D20001F  
Private Sub cmdCancel_Click()
```

```
Unload Me
```

```
End Sub
```

```
'=====
```

```
'Function/Sub Name: Fo rm_Load()
```

```
'Description: This sub resets flags when the form is opened.
```

```
'Input: None
```

```
'Output: None
```

```
'References: None
```

```
'=====
```

```
##ModelId=3B294D20004E  
Private Sub Form_Load()
```

```
Reset global variable for indicating a successful FTP to  
false  
gblnFTPSuccess = False
```

```
Enable buttons  
EnableControls False
```

```
End Sub
```

```
'=====
```

```
'Function/Sub Name: Form_Load()
```

```
'Description: This sub verifies that the FTP is being  
performed on  
'a local server and initiates the FTP connection by  
instantiating  
'an FTP server object. It then downloads the first new  
database file  
'(HFACS.mdf) to the application path. When download of  
the first  
'file is complete, the CallbackCls interface is notified by the  
'FTP server, which in turn executes the download of the next  
file  
'via the GotFileDoNext() sub.
```

```
'Input: None
```

```
'Output: None
```

```

References: None
'=====
##ModelId=3B294D20007D
Private Sub cmdConnect_Click()

    On Error GoTo vbErrorHandler

    If HFACSMMain.gStrServerName <> "(local)" Then
        MsgBox "You can only perform an update when logged
into the " & _
        "'(local)' server.", _
        vbExclamation + vbOKOnly, "Can't Update"
        GoTo ExitSub
    End If

    Screen.MousePointer = vbHourglass
    frmFtpUpdate.lblAction.Caption = "Initializing connection
..."
    frmFtpUpdate.lblAction.Refresh

    ' ask the FTP server to create an FTP object
    Set oDoFTPThread = New HFACSFTP.cFTP

    'Connect
    oDoFTPThread.Connect txtServer.Text, txtUser.Text,
txtPassword.Text
    frmFtpUpdate.lblAction.Caption = "Downloading .mdf file
(this " & _
    "'could take awhile) . . ."
    frmFtpUpdate.lblAction.Refresh

    'Disable buttons
    EnableControls oDoFTPThread.Connected

    'Download the file
    frmFtpUpdate.gifDownloading.Visible = True 'Show
animated GIF
    frmFtpUpdate.gifDownloading.Play
    'Add a '\' to the end of a path entry if they left it off
    If (Len(txtPath.Text) > 1) And (Right(txtPath.Text, 1) <>
"\") _
    Then txtPath.Text = txtPath.Text & "\"
    'Remove a leading '\' from a path entry.
    If Left(txtPath.Text, 1) = "\" Then txtPath.Text = _
        Right(txtPath.Text, Len(txtPath.Text) - 1)

    'Set flag for callback function
    gIntCounter = 1

    'Download the first file
    oDoFTPThread.StartGetFTP txtPath.Text & _
        HFACSMMain.gStrDatabaseFileName,
HFACSMMain.gStrAppPath & _
        "UPDATE-" & HFACSMMain.gStrDatabaseFileName, _
        ftBinary, New CallbackCls

ExitSub:

Exit Sub

vbErrorHandler:
    Screen.MousePointer = vbDefault
    frmFtpUpdate.gifDownloading.Visible = False 'Hide
animated GIF
    If Err.Number = -2147219498 Then
        'This traps a bad path entry

```

```

        MsgBox "Can't find that path on the FTP server.",
vbOKOnly, "Error"
        Resume ExitSub
    Else 'Unknown error
        MsgBox "An error occurred attempting FTP." &
Chr(13) & Chr(13) & _
        "The detailed error message is: " & Err.Description &
Chr(13) & _
        Chr(13) & "Error Number: " & Err.Number, _
        vbOKOnly + vbCritical, "FTP Error"
    End If
    Resume ExitSub

End Sub

'=====
Function/Sub Name: GotFileDoNext()
'
'Description: This sub downloads the second new database
file
'(HFACS_log.ldf) to the application path. When download
of the
'file is complete, the CallbackCls interface is notified by the
'FTP server, which in turn executes the installation of the 2
files
'via the GotFileLast() sub.
'
'Input: None
'
'Output: None
'

References: None
'=====
##ModelId=3B294D2000AB
Friend Sub GotFileDoNext()

    On Error GoTo vbErrorHandler

    ' ask the FTP server to create a new FTP object
    oDoFTPThread.DisConnect
    'Set oDoFTPThread = Nothing
    'Set oDoFTPThread = New HFACSFTP.cFTP

    'Connect
    oDoFTPThread.Connect txtServer.Text, txtUser.Text,
txtPassword.Text

    'Set flag for callback function
    gIntCounter = 2

    'Download the second file
    frmFtpUpdate.lblAction.Caption = "Downloading _log.ldf
file . . ."
    frmFtpUpdate.lblAction.Refresh
    sTempJustTheFileName =
Left(HFACSMMain.gStrDatabaseFileName, _
        (Len(HFACSMMain.gStrDatabaseFileName) - 4)) &
        "_log.ldf"
    oDoFTPThread.StartGetFTP txtPath.Text &
sTempJustTheFileName, _
        HFACSMMain.gStrAppPath & "UPDATE2-" &
sTempJustTheFileName, ftBinary, New CallbackCls

ExitSub:

Exit Sub

vbErrorHandler:

```

```

Screen.MousePointer = vbDefault
frmFtpUpdate.gifDownloading.Visible = False 'Hide
animated GIF
MsgBox "An error occurred attempting FTP." & Chr(13)
& Chr(13) & _
"The detailed error message is: " & Err.Description &
Chr(13) & _
Chr(13) & "Error Number: " & Err.Number, _
vbOKOnly + vbCritical, "Error Attempting FTP"
Unload Me

End Sub

'=====
'Function/Sub Name: GotFileLast()
'
'Description: This sub performs the actual update, updating
'the form to show status as it progresses.
'
'Input: None
'
'Output: None
'
'References: None
'=====
'##ModelId=3B294D2000DA
Friend Sub GotFileLast()

    On Error GoTo vbErrorHandler

    'Destroy connection to the FTP server
    oDoFTPThread.DisConnect
    Set oDoFTPThread = Nothing

    frmFtpUpdate.gifDownloading.Visible = False 'Hide
    animated GIF

    frmFtpUpdate.lblAction.Caption = "File downloaded.
    Creating backups . . ."
    frmFtpUpdate.lblAction.Refresh
    DoEvents 'Redraw screen and check for events
    cmdCancel.Enabled = False

    'Declare an object for hard disk file manipulation
    Dim FSO As Scripting.FileSystemObject
    Set FSO = CreateObject("Scripting.FileSystemObject")
    'Back up the existing hfacs.mdf and rename the
    downloaded file to hfacs.mdf
    On Error GoTo 0
    On Error Resume Next 'Turn off error checking for the
    disk manipulation
    FSO.DeleteFile HFACSMMain.gStrAppPath &
    HFACSMMain.gStrDatabaseFileName, True
    FSO.CopyFile HFACSMMain.gStrAppPath & "UPDATE-"
    & _
    HFACSMMain.gStrDatabaseFileName, _
    HFACSMMain.gStrAppPath &
    HFACSMMain.gStrDatabaseFileName, True
    FSO.DeleteFile HFACSMMain.gStrAppPath & "UPDATE-"
    & _
    HFACSMMain.gStrDatabaseFileName, True
    FSO.DeleteFile HFACSMMain.gStrAppPath &
    sTempJustTheFileName, True
    FSO.CopyFile HFACSMMain.gStrAppPath & "UPDATE2-"
    & _

```

```

sTempJustTheFileName, HFACSMMain.gStrAppPath &
sTempJustTheFileName, True
FSO.DeleteFile HFACSMMain.gStrAppPath & "UPDATE2-"
& sTempJustTheFileName, True
On Error GoTo 0
On Error GoTo vbErrorHandler

'Install the new File
frmFtpUpdate.lblAction.Caption = "Installing . . ."
frmFtpUpdate.lblAction.Refresh
DoEvents 'Redraw screen and check for events

***** Put the Code here
Dim bTestSuccess As Boolean
bTestSuccess = True
bTestSuccess =
Constructors.New_MSDE(HFACSMMain.gStrUID, _
    HFACSMMain.gStrPWD, _
    HFACSMMain.gStrServerName, _
    HFACSMMain.gStrDatabaseFileName, _
    HFACSMMain.gStrDatabaseName, _
    HFACSMMain.gStrAppPath, _
    HFACSMMain.gStrAutoLogon, _
    HFACSMMain.gStrFirstRun, _
    HFACSMMain.gStrNTauth, _
    HFACSMMain.gStrTypeDB)

oMSDE.restartMSDE
DoEvents 'Redraw screen and check for events
frmFtpUpdate.lblAction.Caption = "Dropping old database
. . ."
frmFtpUpdate.lblAction.Refresh
Me.Refresh

If oMSDE.dropDB <> True Then
    gblnFTPSuccess = False
    GoTo ExitSub
End If
DoEvents 'Redraw screen and check for events
frmFtpUpdate.lblAction.Caption = "Attaching new
database . . ."
frmFtpUpdate.lblAction.Refresh

If oMSDE.StartAndCopy <> True Then
    gblnFTPSuccess = False
    GoTo ExitSub
End If
DoEvents 'Redraw screen and check for events

frmFtpUpdate.lblAction.Caption = "Finishing up . . ."
frmFtpUpdate.lblAction.Refresh
Screen.MousePointer = vbDefault

MsgBox "The HFACS file was successfully installed." &
Chr(13) _
& Chr(13) & "HFACS will now re-initialize.", _
vbInformation + vbOKOnly, "Finished"
gblnFTPSuccess = True

ExitSub:
Set oMSDE = Nothing
Set FSO = Nothing
Unload Me

Exit Sub

vbErrorHandler:
Screen.MousePointer = vbDefault

```

```

frmFtpUpdate.gifDownloading.Visible = False 'Hide
animated GIF
If Err.Number = -2147219498 Then
    'This traps a bad path entry
    frmFtpUpdate.lblAction.Caption = "Can't find that path
on the " & _
        "FTP server or the connection was lost."
    frmFtpUpdate.lblAction.Refresh
    MsgBox "Can't find that path on the FTP server.",
vbOKOnly, "Error"
    Resume ExitSub
Else 'Unknown error
    frmFtpUpdate.lblAction.Caption = Err.Description
    frmFtpUpdate.lblAction.Refresh
    MsgBox "An error occurred trying to install the files
after" & _
        "download. Verify that you have adequate
permissions to " & _
        "perform this update." & Chr(13) & Chr(13) & _
        "The detailed error message is: " & Err.Description &
Chr(13) _
        & Chr(13) & "Error Number: " & Err.Number, _
vbOKOnly + vbCritical, "Error Installing Files"
End If
Resume ExitSub

```

End Sub

```

'=====
'Function/Sub Name: cmdDisconnect_Click()
'
'Description: This sub performs disconnect from the FTP
server
'when it is enabled. It is not enabled except during
development.
'
'Input: None
'
'Output: None
'
'References: None
'=====

```

```

###ModelId=3B294D200109
Private Sub cmdDisconnect_Click()

```

```

On Error GoTo vbErrorHandler

```

```

'Disconnect from the FTP Server

```

```

    oDoFTPThread.DisConnect
    EnableControls oDoFTPThread.Connected

```

```

Exit Sub

```

```

vbErrorHandler:
    MsgBox Err.Description

```

End Sub

```

'=====
'Function/Sub Name: EnableControls()
'
'Description: This sub performs dynamically enables/disables
buttons
'on the form based upon the connection state of the FTP
server.
'
'Input:
' bConnected - Boolean value indicating that the server is
connected or disconnected.
'
'Output: None
'
'References: None
'=====

```

```

###ModelId=3B294D200138
Private Sub EnableControls(ByVal bConnected As Boolean)

```

```

    txtServer.Enabled = Not (bConnected)
    txtPath.Enabled = Not (bConnected)
    txtUser.Enabled = Not (bConnected)
    txtPassword.Enabled = Not (bConnected)
    cmdConnect.Enabled = Not (bConnected)
    cmdDisconnect.Enabled = bConnected

```

End Sub

```

'=====
'Function/Sub Name: Form_Unload()
'
'Description: This sub performs cleanup operations,
ensuring all
'objects are destroyed when the form is closed.
'
'Input:
' Cancel - Determines if form is unloaded or hidden
'
'Output: None
'
'References: None
'=====

```

```

###ModelId=3B294D2001B5
Private Sub Form_Unload(Cancel As Integer)

```

```

    On Error Resume Next
    'Make sure the FTP server is disconnected and destroy it.
    If oDoFTPThread.Connected = True Then
        oDoFTPThread.DisConnect
        Set oDoFTPThread = Nothing

```

End Sub

## **FORMCLASS-ODBLogon**

Option Explicit

```
#####  
' FORM DESCRIPTION  
#####  
'Class Name: frmODBLogon.frm  
'  
'Author: Pat Flanders & Scott Tufts  
'  
'This class is responsible for a prompted logon. I provides  
the  
'capability to query a user for logon parameters and test thier  
'validity against a given instance of a SQL Server.  
'  
'References:  
' - Microsoft Data Formating Object Library 6.0  
' - GIF89 1.0 (For animated GIFs on Forms)  
'  
' NOTE: See function headers for internal component  
references.  
#####
```

```
*****  
' PROPERTIES  
*****
```

```
'Warning flag indicating that the database needs to be  
installed on  
'the local server.  
'##ModelId=3B294D050203  
Private bWarningFlag As Boolean
```

```
*****  
' FUNCTIONS  
*****
```

```
=====
```

```
'Function/Sub Name: chkUseNTAuth_Click()  
'  
'Description: This sub updates form properties when the user  
clicks  
'the "Use NT Authentication" check box. It "grey's out" the  
username  
'and password text boxes and makes them unavailable for  
update.  
'  
'Input: None  
'  
'Output: None  
'  
'References: None  
=====
```

```
'##ModelId=3B294D050280  
Private Sub chkUseNTAuth_Click()
```

```
    If chkUseNTAuth.Value = 1 Then  
        frmODBLogon.txtUID = ""
```

```
        frmODBLogon.txtUID.Enabled = False  
        frmODBLogon.txtUID.BackColor = &H8000000F  
        frmODBLogon.txtUID.Refresh  
        frmODBLogon.txtPWD = ""  
        frmODBLogon.txtPWD.Enabled = False  
        frmODBLogon.txtPWD.BackColor = &H8000000F  
        frmODBLogon.txtPWD.Refresh  
    Else  
        frmODBLogon.txtUID = HFACSMMain.gStrUID  
        frmODBLogon.txtUID.Enabled = True  
        frmODBLogon.txtUID.BackColor = &H80000009  
        frmODBLogon.txtUID.Refresh  
        frmODBLogon.txtPWD = ""  
        frmODBLogon.txtPWD.Enabled = True  
        frmODBLogon.txtPWD.BackColor = &H80000009  
        frmODBLogon.txtPWD.Refresh  
    End If
```

End Sub

```
=====
```

```
'Function/Sub Name: cmdCancel_Click()  
'
```

```
'Description: This sub closes the form  
'
```

```
'Input: None  
'
```

```
'Output: None  
'
```

```
'References: None  
=====
```

```
'##ModelId=3B294D0502AF  
Private Sub cmdCancel_Click()
```

Unload Me

End Sub

```
=====
```

```
'Function/Sub Name: cmdOk_Click()  
'
```

```
'Description: This sub combines the functionality of testing  
the  
'connection with the user supplied paramaters and, if the  
parameters  
'are valid, updating the pertinent global variables to enable  
'other component class intances to function (e.g. to update the  
'ini file with new settings).  
'
```

```
'Input: None  
'
```

```
'Output: None  
'
```

```
'References:  
' - Constructors.bas  
' - MSDE.cls  
' - HFACSMMain.bas  
=====
```

```
'##ModelId=3B294D0502DE  
Private Sub cmdOk_Click()
```

On Error GoTo StartError



```

'Only proceed with the update if all tests are passed
If testNewConn() = True Then

    Screen.MousePointer = 11
    Dim bResponseStartAndCopy As Boolean

    'Check if NT Auth should be used
    Dim sNTAuth As String
    If chkUseNTAuth.Value = 1 Then
        sNTAuth = "T"
    Else
        sNTAuth = "F"
    End If

    'Create an instance of MSDE copy the database if
    needed
    bResponseStartAndCopy =
Constructors.New_MSDE(txtUID.Text, _
    txtPWD.Text, txtServer.Text,
    HFACSMMain.gStrDatabaseFileName, _
    txtDatabase.Text, HFACSMMain.gStrAppPath, _
    HFACSMMain.gStrAutoLogon,
    HFACSMMain.gStrFirstRun, sNTAuth, _
    HFACSMMain.gStrTypeDB)
    bResponseStartAndCopy = oMSDE.StartAndCopy
    Set oMSDE = Nothing

    If bResponseStartAndCopy = True Then
        'Set global variables to new values
        gStrUID = frmODBLLogon.txtUID.Text
        gStrPWD = frmODBLLogon.txtPWD.Text
        gStrServerName = frmODBLLogon.txtServer.Text
        gStrDatabaseName = frmODBLLogon.txtDatabase
        If frmODBLLogon.txtServer.Text = "(local)" And _
            frmODBLLogon.txtPWD = "" Then
            gStrAutoLogon = "T"
        Else
            gStrAutoLogon = "F"
        End If
        If chkUseNTAuth.Value = 1 Then
            gStrNTAuth = "T"
        Else
            gStrNTAuth = "F"
        End If
        Screen.MousePointer = 0
        MsgBox "Successfully connected to server: " &
txtServer.Text, _
            vbInformation + vbOKOnly, "Connected"
        gblnPromptedLogonSuccess = True
        Unload Me
    Else
        Screen.MousePointer = 0
        MsgBox "There is an unknown problem with this
connection." & _
            Chr(13) & Chr(13), vbExclamation + vbOKOnly, _
            "Connection Refused"
        gblnPromptedLogonSuccess = False
        Unload Me
    End If
End If

ExitSub:
    Screen.MousePointer = 0
Exit Sub

StartError:
    Screen.MousePointer = 0

    MsgBox "An unknown error occured in frmODBLLogon at
method" & _
        " cmdOK_Click." & Chr(13) & Chr(13) & _
        "The detailed error message is: " & _
        Err.Description & Chr(13) & Chr(13) & _
        "Error Number: " & Err.Number, _
        vbOKOnly + vbCritical, "Error"
    Resume ExitSub

End Sub

'=====
Function/Sub Name: cmdTest_Click()
'
'Description: This sub calls the testNewConn() function and
returns
'an appropriate message to the user.
'Input: None
'
'Output: None
'
'References: None
'=====
'##ModelId=3B294D05030D
Private Sub cmdTest_Click()

    Dim bTestResults As Boolean 'Placeholder for test results
    bTestResults = testNewConn() 'Run a test
    If bTestResults = True And bWarningFlag = False Then
        MsgBox "Connection test succeeded!", vbInformation +
vbOKOnly, _
            "Test Succeeded"
    End If
    If bTestResults = True And bWarningFlag = True Then
        MsgBox "The database you specified is not installed." &
_
            Chr(13) & Chr(13) & "If you proceed with this
connection, " & _
            "you must have ADMINISTRATOR priveleges to the
this machine " & _
            "and the database so that the database can be " & _
            "automatically installed.", vbExclamation, "Warning"
    End If

End Sub

'=====
Function/Sub Name: Form_Load()
'
'Description: This sub sets the states of the form controls
(visible/
'not visible and enabled/disabled) based upon current global
variable
'settings.
'
'Input: None
'
'Output: None
'
'References:
' - Constructors.bas
' - HFACSMMain.bas
'=====
'##ModelId=3B294D05033C
Private Sub Form_Load()

```

```

'Ensure the logon success flag is reset to false
gblnPromptedLogonSuccess = False

'Set initial value to false
gblnNoCopyNeeded = False

'Populate the combobox
Dim i As Integer
'Use the SQL DMO Application Object to find the
available SQL Servers
Dim oSQLServerDMOApp As sqldmo.Application
Set oSQLServerDMOApp = New sqldmo.Application

txtServer.AddItem "(local)"
'Turn off error checking incase there are no servers
detected
On Error Resume Next
Dim namX As NameList
Set namX =
oSQLServerDMOApp.ListAvailableSQLServers
For i = 1 To namX.Count
    If namX.Item(i) <> "(local)" Then
        txtServer.AddItem namX.Item(i)
    End If
Next
'Show top server
txtServer.ListIndex = 0

txtServer.Text = HFACSMMain.gStrServerName
'Populate the other text boxes
txtUID.Text = HFACSMMain.gStrUID
txtDatabase.Text = HFACSMMain.gStrDatabaseName
txtPWD.Text = HFACSMMain.gStrPWD
If HFACSMMain.gStrNTAuth = "T" Then
    chkUseNTAuth.Value = 1
    frmODBlogon.txtUID.Enabled = False
    frmODBlogon.txtUID.BackColor = &H8000000F
    frmODBlogon.txtUID.Refresh
    frmODBlogon.txtPWD = ""
    frmODBlogon.txtPWD.Enabled = False
    frmODBlogon.txtPWD.BackColor = &H8000000F
    frmODBlogon.txtPWD.Refresh
End If
End Sub

'=====
'Function/Sub Name: testNewConn()
'
'Description: This sub tests the validity of the user specified
'connection values by attempting to start and connect to the
'server. Upon successful connection to the server specified, it
'verifies existence of the HFACS database on that server.
'
'Input: None
'
'Output: None
'
'References:
' - Constructors.bas
' - MSDE.cls
' - HFACSMMain.bas
'=====

```

```

'##ModelId=3B294D05036B
Private Function testNewConn() As Boolean

    On Error GoTo StartError

    Screen.MousePointer = 11
    frmODBlogon.lblAction.Visible = True 'Show the
connect message
    frmODBlogon.lblAction.Refresh
    frmODBlogon.gifNetwork.Visible = True 'Show animated
GIF
    frmODBlogon.gifNetwork.Play

    'For some reason, if a user enters a ';' it messup up the
'logon . . . so remove them.
txtUID.Text = Replace(txtUID.Text, ";", "")

'Set Flags
bWarningFlag = False
testNewConn = False
Dim bResponseServer As Boolean
Dim bResponseDatabase As Boolean

'Check if NT Auth should be used
Dim sNTAuth As String
If Me.chkUseNTAuth.Value = 1 Then
    sNTAuth = "T"
Else
    sNTAuth = "F"
End If

'For some reason, a logon will work using NTAuth, even if
the
'checkbox isn't checked,
'So this code will stop that.
If chkUseNTAuth.Value = 0 And txtUID.Text = "" And
txtPWD.Text = "" Then
    MsgBox "Invalid User ID or Password.", vbCritical,
"Connection Failed"
    testNewConn = False
    GoTo TestConnFailure
End If

'Test the ability to start and connect to an MSDE or SQL
server
bResponseServer =
Constructors.New_MSDE(txtUID.Text, txtPWD.Text, _
txtServer.Text, HFACSMMain.gStrDatabaseFileName, _
txtDatabase.Text, HFACSMMain.gStrAppPath, _
HFACSMMain.gStrAutoLogon,
HFACSMMain.gStrFirstRun, _
sNTAuth, HFACSMMain.gStrTypeDB)
bResponseServer = oMSDE.startMSDE

'Check for a remote connection to SQL 2k . . . if this is a
'remote connection attempt, and it works, then no copy is
'needed, so just quit this function and return true.
If gblnNoCopyNeeded = True Then testNewConn = True:
GoTo ExitSub

If bResponseServer = True Then
    'Now test for the existance of the database
    bResponseDatabase = oMSDE.databaseExists

    If bResponseDatabase = True Then
        testNewConn = True
    Else

```

```

        'Finally, test to see if the SQL server is local or
remote.
        If txtServer.Text = "(local)" Then
            testNewConn = True
            bWarningFlag = True
        Else
            Screen.MousePointer = 0
            MsgBox "The server you specified exists, but it is
not" & _
                " on the local machine and the database you
specified" & _
                " is not installed." & Chr(13) & Chr(13) & _
                "This program cannot create a database on a
machine " & _
                "other than the local machine.", vbCritical, _
                "Connection Failed"
            testNewConn = False
        End If
    End If
End If

ExitSub:
    Screen.MousePointer = 0
    Set oMSDE = Nothing

```

```

TestConnFailure:
    Screen.MousePointer = 0
    frmODBLgon.lblAction.Visible = False 'Hide the connect
message
    frmODBLgon.lblAction.Refresh
    frmODBLgon.gifNetwork.Visible = False 'Hide animated
GIF

```

Exit Function

```

StartError:
    Screen.MousePointer = 0
    MsgBox "Destination host unreachable. The server may
not " & _
        "be started or you may have to build a System DSN." &
Chr(13) & Chr(13) & _
        "The detailed error message is: frmODBLgon - " &
Err.Description & _
        Chr(13) & Chr(13) & "Error Number: " & Err.Number,
    -
        vbOKOnly + vbCritical, "Connection Failed"
    Resume ExitSub

```

End Function

## **FORMCLASS-Wait**

Option Explicit

```
'#####  
'          FORM DESCRIPTION  
'#####  
'Class Name: frmWait.frm  
'  
'Author: Pat Flanders & Scott Tufts  
'  
'This class is responsible for showing a status bar capable of  
'pausing the number of seconds specified by  
'HFACSMMain.gIntTimeToWait and displaying the message  
'contained  
'in HFACSMMain.gStrTextMessage.  
'  
'References:  
' - Microsoft Windows Common Controls 6.0  
'  
'NOTE: See function headers for internal component  
references.  
'#####
```

```
*****  
'          FUNCTIONS  
*****
```

```
'=====
```

Function/Sub Name: Form\_GotFocus()

Description: This sub reads the values contained in the global variables to determine how long to show itself and what message to display.

Input: None

Output: None

References:  
- HFACSMMain.bas

```
'=====
```

##ModelId=3B294D0F0138

Private Sub Form\_GotFocus()

```
    guaStatus.Value = 0  
    guaStatus.Max = HFACSMMain.gIntTimeToWait  
  
    Screen.MousePointer = vbHourglass  
  
    Dim PauseTime  
    Dim Start  
    Dim i As Integer  
  
    'Retrieve the duration from the global variable  
    PauseTime = HFACSMMain.gIntTimeToWait  
  
    Start = Timer ' Set start time.  
    Do While Timer < Start + PauseTime  
        guaStatus.Value = Abs(Timer - Start)  
        DoEvents ' Yield to other processes.  
    Loop  
    Screen.MousePointer = vbDefault  
    Unload Me
```

End Sub

```
'=====
```

Function/Sub Name: Form\_Load()

Description: This sub reads the values contained in the global variables to determine the message to display on the form.

Input: None

Output: None

References:  
- HFACSMMain.bas

```
'=====
```

##ModelId=3B294D0F0167

Private Sub Form\_Load()

frmWait.lblAction.Caption =  
HFACSMMain.gStrTextMessage

End Sub

## **MODULE-Constructors**

Option Explicit

```
#####
'      MODULE DESCRIPTION
#####
'Module Name: Constructors.bas

'Author: Pat Flanders & Scott Tufts

'This module defines functions that pair creation of new
object
'instances using the reusable global objects defined in
HFACSMain
'with a call to an init() function of the associated class. In
this
'manner, these functions can act as psuedo-constructors that
are
'capable of passing arguments -- a feature not available in
Visual
Basic 6.0.
#####
```

```
*****
'      FUNCTIONS
*****
```

```
=====
Function/Sub Name: New_INIFile()

'Description: This function acts as a psuedo-constructor. It
'creates a new INIFile object and calls the INIFile.init()
function,
'passing desired parameters to ensure a consistent state.
'
'Input:
' sFileName - String value of representing the name of the
'           the .ini file to manipulate.
'
'Output: None
'
'References: INIFile.cls
=====
##ModelId=3B294D140138
Public Function New_INIFile(sFileName As String)

    Set oINIFile = New INIFile

    'Set the INIFile class instance to always use the global ini
    'filename for read/write operations
    oINIFile.Init gINIFILENAME

End Function
```

```
=====
Function/Sub Name: New_INIFileController()
'
'Description: This function acts as a psuedo-constructor. It
'creates a new INIFileController object and calls the
INIFileController.init() function, passing desired parameters
'to ensure a consistent state.
```

```
'
'Input: Currently, none. Future implementation may require
'       parameters, so this code remains.
'
'Output: None
'
'References: INIFile.cls
=====
##ModelId=3B294D140177
Public Function New_INIFileController()

    Set oINIFileController = New INIFileController
    oINIFileController.Init
```

End Function

```
=====
Function/Sub Name: New_HFACSCConnection()
'
'Description: This function acts as a psuedo-constructor. It
'creates a new HFACSCConnection object and calls the
'HFACSCConnection.init() function, passing desired
parameters
'to ensure a consistent state.
'
'Input:
' sUser      - The user ID
' sPassword  - The user password
' sSvrName   - The name of the MSDE or SQL Server
' sMDFName   - The name of the .mdf file containing the
'             database.
' sDBName    - The name of the database
' sInstDirectory - The application path
' sAutomaticLogon - Toggle to log on with/without prompt
' sFirstRunCheck - Toggle for determining if this is the first
run
'             after an update.
' sNTAuth    - Toggle for determining if NT
authentication
'             should be used for logon attempts.
' sTypeDB    - The type of DB this program will
represent
'             (mil, civ, or both).'
'
'Output: None
'
'References: HFACSCConnection.cls
=====
```

```
##ModelId=3B294D1401A5
Public Function New_HFACSCConnection(Optional sUser As
String, Optional sPassword As String, Optional sSvrName As
String, Optional sMDFName As String, Optional sDBName
As String, Optional sInstDirectory As String, Optional
sAutomaticLogon As String, Optional sFirstRunCheck As
String, Optional sNTAuth As String, Optional sTypeDB As
String)
```

```
    Set oHFACSCConnection = New HFACSCConnection

    'Set the MSDE class instance default values
    If IsMissing(sUser) Then sUser = gStrUID
    If IsMissing(sPassword) Then sPassword = gStrPWD
    If IsMissing(sSvrName) Then sSvrName =
gStrServerName
```

```

    If IsMissing(sMDFName) Then sMDFName =
gStrDatabaseFileName
    If IsMissing(sDBName) Then sDBName =
gStrDatabaseName
    If IsMissing(sInstDirectory) Then sInstDirectory =
gStrAppPath
    If IsMissing(sAutomaticLogon) Then sAutomaticLogon =
gStrAutoLogon
    If IsMissing(sFirstRunCheck) Then sFirstRunCheck =
gStrFirstRun
    If IsMissing(sNTAuth) Then sNTAuth = gStrNTAuth
    If IsMissing(sTypeDB) Then sTypeDB = gStrTypeDB

oHFACSConnection.Init sUser, _
    sPassword, _
    sSvrName, _
    sMDFName, _
    sDBName, _
    sInstDirectory, _
    sAutomaticLogon, _
    sFirstRunCheck, _
    sNTAuth, _
    sTypeDB

End Function

'=====
Function/Sub Name: New_MSDE()
'
'Description: This function acts as a psuedo-constructor. It
'creates a new MSDE object and calls the MSDE.init()
'function,
'passing desired parameters to ensure a consistent state.
'
'Input:
' sUser      - The user ID
' sPassword   - The user password
' sSvrName    - The name of the MSDE or SQL Server
' sMDFName    - The name of the .mdf file containing the
'               database.
' sDBName     - The name of the database
' sInstDirectory - The application path
' sAutomaticLogon - Toggle to log on with/without prompt
' sFirstRunCheck - Toggle for determining if this is the first
run
'               after an update.
' sNTAuth     - Toggle for determining if NT
authentication
'               should be used for logon attempts.
' sTypeDB     - The type of DB this program will
represent

```

```

'               (mil, civ, or both).'

'Output: None
'
'References: MSDE.cls
'=====
'##ModelId=3B294D140290
Public Function New_MSDE(Optional sUser As String,
Optional sPassword As String, Optional sSvrName As String,
Optional sMDFName As String, Optional sDBName As
String, Optional sInstDirectory As String, Optional
sAutomaticLogon As String, Optional sFirstRunCheck As
String, Optional sNTAuth As String, Optional sTypeDB As
String)

    Set oMSDE = New MSDE

    'Set the MSDE class instance default values.
    'Notice that password remains "" if it is missing. This
forces
    'a prompted logon.
    If IsMissing(sUser) Then sUser = gStrUID
    If IsMissing(sPassword) Then sPassword = ""
    If IsMissing(sSvrName) Then sSvrName =
gStrServerName
    If IsMissing(sMDFName) Then sMDFName =
gStrDatabaseFileName
    If IsMissing(sDBName) Then sDBName =
gStrDatabaseName
    If IsMissing(sInstDirectory) Then sInstDirectory =
gStrAppPath
    If IsMissing(sAutomaticLogon) Then sAutomaticLogon =
gStrAutoLogon
    If IsMissing(sFirstRunCheck) Then sFirstRunCheck =
gStrFirstRun
    If IsMissing(sNTAuth) Then sNTAuth = gStrNTAuth
    If IsMissing(sTypeDB) Then sTypeDB = gStrTypeDB

oMSDE.Init sUser, _
    sPassword, _
    sSvrName, _
    sMDFName, _
    sDBName, _
    sInstDirectory, _
    sAutomaticLogon, _
    sFirstRunCheck, _
    sNTAuth, _
    sTypeDB

End Function

```

## **MODULE-HFACSMain**

Option Explicit

```
#####
'
'      MODULE DESCRIPTION
'#####
'Module Name: HFACSMMain.bas
'
'Author: Pat Flanders & Scott Tufts
'
'This module is accessible to all classes and forms in the
project.
'It contains declarations for all global variables used to pass
'values between forms and instances of classes.
'
'References For The Entire Component:
' - Microsoft Data Formatting Object Library 6.
' - Microsoft ActiveX Data Objects 2.5 Library
' - Microsoft SQLDMO Object Library
' - Microsoft Scripting Runtime
' - GIF89 1.0 (For animated GIFs on Forms)
' - The HFACSFtp.exe ftp server.
'#####
```

```
*****
'
'      GLOBAL VARIABLES
'*****
```

```
'This variable is used by HFACSMMain.Main() for initializing
the entire
'component. It is required for all compiled DLLs, but not
used for
'else.
'##ModelId=3B294CE9034B
Public gdatServerStarted As Date
```

```
'Constant variable to hold the name of the .ini file.
'##ModelId=3B294CEA007D
Global Const gINIFILENAME As String = "hfacs"
```

```
'Reusable object variables. These variables are used over and
over
'by classes and forms. They are created and destroyed within
the
'same function whenever possible.
```

```
'-----
'Reusable object variable for the INI file
'##ModelId=3B294CEB0010
Global oINIFile As INIFile
```

```
'Reusable object variable for the INI file control class
'##ModelId=3B294CEC02C1
Global oINIFileController As INIFileController
```

```
'Reusable object variable for the HFACSCConnection class
'##ModelId=3B294CEE01A5
Global oHFACSCConnection As HFACSCConnection
```

```
'##ModelId=3B294CF0006E
```

```
Global oMSDE As MSDE 'Reusable object variable for the
MSDE Class
```

```
'Reusable object variable for the UpdateController Class
'##ModelId=3B294CF1032D
Global oUpdateController As UpdateController
```

```
'Variable to hold the path to the Windows system directory
'##ModelId=3B294CF202CE
Global gStrFileName As String 'The name of the system
directory
```

```
'INI file declarations. Each of these variables represents an
entry
'in the .ini file. These values are the core of much of the
'of this component and as such are visible to all forms and
classes.
```

```
'-----
'##ModelId=3B294CF2031C
Global gStrUID As String 'The user ID
```

```
'##ModelId=3B294CF2036B
Global gStrPWD As String 'The user password
```

```
'##ModelId=3B294CF203A9
Global gStrServerName As String 'The name of the MSDE
or SQL Server
```

```
'##ModelId=3B294CF3001F
Global gStrDatabaseFileName As String 'The name of the
mdf
```

```
'##ModelId=3B294CF3006D
Global gStrDatabaseName As String 'The name of the
database
```

```
'##ModelId=3B294CF300BB
Global gStrAppPath As String 'The application path
```

```
'##ModelId=3B294CF30109
Global gStrAutoLogon As String 'Toggle to logon without
prompt
```

```
'Toggle for determining the first time the program has been
run.
'##ModelId=3B294CF30157
Global gStrFirstRun As String
```

```
'Toggle for determining if NT authentication should be used
for
'logon attempts.
'##ModelId=3B294CF301A5
Global gStrNTauth As String
```

```
'The type of DB this program will represent (mil, civ, or
both).
'##ModelId=3B294CF301F4
Global gStrTypeDB As String
```

```
'Global variable to hold the value of the current
connectionstring
'##ModelId=3B294CF30242
```

Global gTheConnectionString As String

'Global variable to hold the value of the SQL Server  
subdirectory  
Global gSQLServerPath As String

'Flags for passing success or failure of form operations

'-----  
'Flag a success/failure of a prompted logon  
##ModelId=3B294CF30290  
Global gblnPromptedLogonSuccess As Boolean

'Flag a success/failure of an FTP update attempt  
##ModelId=3B294CF302DE  
Global gblnFTPSuccess As Boolean

'Flags for passing strings and integer values between forms

'-----  
'Message for label on frmWait. Allows you to change the  
message from  
'any location in this component.  
##ModelId=3B294CF3032C  
Global gStrTextMessage As String

'Amount of time for frmWait to count. Allows you to set the  
number  
'of seconds for frmWait to actually wait.  
##ModelId=3B294CF3037A  
Global gIntTimeToWait As Integer

'Reusable variable for counters throughout the component  
##ModelId=3B294CF303C8  
Global gIntCounter As Integer

'Flag for indicating no copy is necessary. This is required  
'when making a connection to a remote host because the SQL  
Server  
'2000 version of SQLDMO won't connect to a remote host.  
To work  
'around this, an ADO connection is attempted. If an ADO  
connection  
'succeeds, then the database exists on the server being  
connected  
'to, so no copy is needed . . . and this flag is set.  
##ModelId=3B294CF4002E  
Global gblnNoCopyNeeded As Boolean

\*\*\*\*\*  
' GLOBAL UTILITY FUNCTIONS

\*\*\*\*\*

'-----  
'Function/Sub Name: Main()

'  
'Description: This code is executed when the component  
starts, in  
'response to the first object request. It is the "Main"  
procedure  
'responsible for initializing the entire component and is  
required  
'for all compiled DLLs.

'Input: None

'Output: None

'References: None

'-----  
##ModelId=3B294CF4006D  
Sub Main()

gdatServerStarted = Now()  
Debug.Print ""  
Debug.Print "Executing Sub Main . . ."

End Sub

'-----  
'Function/Sub Name: IsOpen()

'  
'Description: Determines if a form is open or not. Useful for  
'determining when screen refreshes are needed.

'Input: String representing the name of the form to be  
checked.

'Output: True if the form is open, otherwise false.

'References: None

'-----  
##ModelId=3B294CF4009C  
Public Function IsOpen(szName As String) As Boolean

IsOpen = (SysCmd(acSysCmdGetObjectState, acForm,  
szName) <> 0)

End Function



## APPENDIX H. CLIPBOARD UTILITY

### *CLASS-clsClipboard*

Option Explicit

#####

' CLASS DESCRIPTION

#####

'Class Name: clsClipboard

'Author: Pat Flanders & Scott Tufts

'Description: The Access 2000 VBA IDE does not allow  
direct access  
'to the "clipboard" object. This class wraps the functionality  
'of parts of the clipboard object in VB 6.0.

'References: None

'NOTE: See function headers for internal component  
references.

#####

\*\*\*\*\*

' FUNCTIONS

\*\*\*\*\*

'=====

'=====

'Function/Sub Name: clipOutLandscape()

'Description: Prints the contents of the Windows clipboard  
Horizontally on a printed page.

'Input: None

'Output: Success or failure.

'References: None

'=====

Public Function clipOutLandscape() As Boolean

On Error GoTo StartError

Printer.Orientation = vbPRORLandscape

Printer.Print " "

Printer.PaintPicture Clipboard.GetData(), 0, 0

Printer.EndDoc

Printer.Orientation = vbPRORPortrait

clipOutLandscape = True

THIS PAGE INTENTIONALLY LEFT BLANK

## APPENDIX I. FTP SERVER

### CLASS-cFTP

```
Option Explicit
'#####
'          CLASS DESCRIPTION
'#####
'Class Name: cFTP
'
'Author: Chris Eastwood, July 1999. Modified by Pat
Flanders &
'Scott Tufts.
'
'Description: Provides FTP functionality in a separate
process.
'This class wraps the functionality of the Win32
WinInet.DLL
'
'References: WinInet.dll via API calls – do not reference
from
'the VB IDE.
'
'#####

'*****
'          DECLARES
'*****

Private Declare Sub Sleep Lib "kernel32" (ByVal
dwMilliseconds As Long)

Dim SaveCBK As cFTPCBK
Dim frmTimer As frmTimer

Private Const MAX_PATH = 260

Private Type FILETIME
    dwLowDateTime As Long
    dwHighDateTime As Long
End Type

Private Type WIN32_FIND_DATA
    dwFileAttributes As Long
    ftCreationTime As FILETIME
    ftLastAccessTime As FILETIME
    ftLastWriteTime As FILETIME
    nFileSizeHigh As Long
    nFileSizeLow As Long
    dwReserved0 As Long
    dwReserved1 As Long
    cFileName As String * MAX_PATH
    cAlternate As String * 14
End Type

Private Const ERROR_NO_MORE_FILES = 18
Private Declare Function InternetFindNextFile Lib
"wininet.dll" Alias "InternetFindNextFileA" (ByVal hFind
As Long, lpvFindData As WIN32_FIND_DATA) As Long

Private Declare Function FtpFindFirstFile Lib "wininet.dll"
Alias "FtpFindFirstFileA" (ByVal hFtpSession As Long,
ByVal lpzSearchFile As String, lpFindFileData As
WIN32_FIND_DATA, ByVal dwFlags As Long, ByVal
dwContent As Long) As Long

Private Declare Function FtpGetFile Lib "wininet.dll" Alias
"FtpGetFileA" (ByVal hFtpSession As Long, ByVal
lpzRemoteFile As String, ByVal lpzNewFile As String,
ByVal fFailIfExists As Boolean, ByVal
dwFlagsAndAttributes As Long, ByVal dwFlags As Long,
ByVal dwContext As Long) As Boolean

Private Declare Function FtpPutFile Lib "wininet.dll" Alias
"FtpPutFileA" (ByVal hFtpSession As Long, ByVal
lpzLocalFile As String, ByVal lpzRemoteFile As String,
ByVal dwFlags As Long, ByVal dwContext As Long) As
Boolean

Private Declare Function FtpSetCurrentDirectory Lib
"wininet.dll" Alias "FtpSetCurrentDirectoryA" (ByVal
hFtpSession As Long, ByVal lpzDirectory As String) As
Boolean
' Initializes an application's use of the Win32 Internet
functions
Private Declare Function InternetOpen Lib "wininet.dll"
Alias "InternetOpenA" (ByVal sAgent As String, ByVal
lAccessType As Long, ByVal sProxyName As String, ByVal
sProxyBypass As String, ByVal lFlags As Long) As Long

' Use registry access settings.
Private Const INTERNET_OPEN_TYPE_DIRECT = 1
Private Const INTERNET_OPEN_TYPE_PROXY = 3
Private Const INTERNET_INVALID_PORT_NUMBER = 0

Private Const FTP_TRANSFER_TYPE_ASCII = &H1
Private Const FTP_TRANSFER_TYPE_BINARY = &H2
Private Const FILE_ATTRIBUTE_NORMAL = &H80
'Added
Private Const INTERNET_FLAG_PASSIVE = &H80000000

Private Declare Function InternetConnect Lib "wininet.dll"
Alias "InternetConnectA" (ByVal hInternetSession As Long,
ByVal sServerName As String, ByVal nServerPort As
Integer, ByVal sUserName As String, ByVal sPassword As
String, ByVal lService As Long, ByVal lFlags As Long,
ByVal lContext As Long) As Long

Private Const ERROR_INTERNET_EXTENDED_ERROR
= 12003

Private Declare Function InternetGetLastResponseInfo Lib
"wininet.dll" Alias "InternetGetLastResponseInfoA"
(lpdwError As Long, ByVal lpzBuffer As String,
lpdwBufferLength As Long) As Boolean

' Type of service to access.
Private Const INTERNET_SERVICE_FTP = 1
```

```

'private Const INTERNET_SERVICE_GOPHER = 2
'private Const INTERNET_SERVICE_HTTP = 3

Private Const INTERNET_FLAG_RELOAD =
&H80000000
Private Const INTERNET_FLAG_KEEP_CONNECTION =
&H400000
Private Const INTERNET_FLAG_MULTIPART =
&H200000

Private Declare Function FtpOpenFile Lib "wininet.dll" Alias
"FtpOpenFileA" (ByVal hFtpSession As Long, ByVal
sFileName As String, ByVal lAccess As Long, ByVal lFlags
As Long, ByVal lContext As Long) As Long
Private Declare Function FtpDeleteFile Lib "wininet.dll"
Alias "FtpDeleteFileA" (ByVal hFtpSession As Long, ByVal
lpszFileName As String) As Boolean

Private Declare Function FtpRenameFile Lib "wininet.dll"
Alias "FtpRenameFileA" (ByVal hFtpSession As Long,
ByVal sExistingName As String, ByVal sNewName As
String) As Boolean

' Closes a single Internet handle or a subtree of Internet
handles.
Private Declare Function InternetCloseHandle Lib
"wininet.dll" (ByVal hInet As Long) As Integer

,
' Our Defined Errors
,

Public Enum errFtpErrors
    errCannotConnect = vbObjectError + 2001
    errNoDirChange = vbObjectError + 2002
    errCannotRename = vbObjectError + 2003
    errCannotDelete = vbObjectError + 2004
    errNotConnectedToSite = vbObjectError + 2005
    errGetFileError = vbObjectError + 2006
    errInvalidProperty = vbObjectError + 2007
    errFatal = vbObjectError + 2008
End Enum

,
' File Transfer types
,

Public Enum FileTransferType
    ftAscii = FTP_TRANSFER_TYPE_ASCII
    ftBinary = FTP_TRANSFER_TYPE_BINARY
End Enum

,
' Error messages
,

Private Const ERRCHANGEDIRSTR As String = "Cannot
Change Directory to %s. It either doesn't exist, or is
protected"
Private Const ERRCONNECTERROR As String = "Cannot
Connect to %s using User and Password Parameters"
Private Const ERRNOCONNECTION As String = "Not
Connected to FTP Site"
Private Const ERRNODOWNLOAD As String = "Couldn't
Get File %s from Server"
Private Const ERRNORENAME As String = "Couldn't
Rename File %s"
Private Const ERRNODELETE As String = "Couldn't Delete
File %s from Server"

```

```

Private Const ERRALREADYCONNECTED As String =
"You cannot change this property while connected to an FTP
server"
Private Const ERRFATALERROR As String = "Cannot get
Connection to WinInet.dll !"

,
' Session Identifier to Windows
,
Private Const SESSION As String = "CGFtp Instance"
,
' Our INET handle
,
Private mlInetHandle As Long
,
' Our FTP Connection Handle
,
Private mlConnection As Long
,
' Standard FTP properties for this class
,
Private msHostAddress As String
Private msUser As String
Private msPassword As String
Private msDirectory As String

' Passed in from HFACS.DLL
Private ServerFileAndPath As String
Private DestinationFileAndPath As String
Private TransferType As FileTransferType

```

```

*****
'
FUNCTIONS
*****

=====
'Function/Sub Name: Initialize()
,
'Description: Opens an Internet session.
,
'Input: None
,
'Output: None
,
'References: None

=====
Private Sub Class_Initialize()
,
' Create Internet session handle

    mlInetHandle = InternetOpen(SESSION,
INTERNET_OPEN_TYPE_DIRECT, vbNullString,
vbNullString, 0)

    If mlInetHandle = 0 Then
        mlConnection = 0
        Err.Raise errFatal, "CGFTP::Class_Initialise",
ERRFATALERROR
    End If

    mlConnection = 0

End Sub

```

```

'=====
'Function/Sub Name: Terminate()
'
'Description: Kills an Internet session.
'
'Input: None
'
'Output: None
'
'References: None
'=====
Private Sub Class_Terminate()
'
' Kill off any connection
'
'   If mlConnection <> 0 Then
'       InternetCloseHandle mlConnection
'   End If
'
' Kill off API Handle
'
'   If mlInetHandle <> 0 Then
'       InternetCloseHandle mlInetHandle
'   End If
'   mlConnection = 0
'   mlInetHandle = 0
'
End Sub

'=====
'Function/Sub Name: Connect()
'
'Description: Connect to the FTP server.
'
'Input:
'   - Host      - IP or name of host
'   - User      - User ID
'   - Password  - Password for FTP logon
'
'Output: Success or failure
'
'References: None
'=====
Public Function Connect(Optional Host As String, _
    Optional User As String, _
    Optional Password As String) As Boolean
'
' Connect to the FTP server
'
On Error GoTo vbErrorHandler

    Dim sError As String
'
' If we already have a connection then raise an error
'
    If mlConnection <> 0 Then
        On Error GoTo 0
        Err.Raise errInvalidProperty, "CGFTP::Connect", "You
are already connected to FTP Server " & msHostAddress
        Exit Function
    End If
'
' Overwrite any existing properties if they were supplied in
the
' arguments to this method
'
    If Len(Host) > 0 Then
        msHostAddress = Host
    End If

    If Len(User) > 0 Then
        msUser = User
    End If

    If Len>Password) > 0 Then
        msPassword = Password
    End If

' Connect !
'
    If Len(msHostAddress) = 0 Then
        Err.Raise errInvalidProperty, "CGFTP::Connect", "No
Host Address Specified!"
    End If

    mlConnection = InternetConnect(mlInetHandle,
msHostAddress, INTERNET_INVALID_PORT_NUMBER,
-
    msUser, msPassword, INTERNET_SERVICE_FTP, 0,
0)
'
' Check for connection errors
'
    If mlConnection = 0 Then
        sError = Replace(ERRCONNECTERROR, "%s",
msHostAddress)
        On Error GoTo 0
        sError = sError & vbCrLf &
GetINETErrorMsg(Err.LastDllError)
        Err.Raise errCannotConnect, "CGFTP::Connect", sError
    End If

    Connect = True

    Exit Function

vbErrorHandler:

    Err.Raise Err.Number, "cFTP::Connect", Err.Description

End Function

'=====
'Function/Sub Name: Disconnect()
'
'Description: Disconnect, only if connected
'
'Input: None
'
'Output: Success or failure
'
'References: None
'=====
Public Function Disconnect() As Boolean
'
' Disconnect, only if connected !
'

```

```

On Error Resume Next
If mlConnection <> 0 Then
    InternetCloseHandle mlConnection
    mlConnection = 0
Else
    'Err.Raise errNotConnectedToSite,
"CGFTP::Disconnect", ERRNOCONNECTION
End If
msHostAddress = ""
msUser = ""
msPassword = ""
msDirectory = ""

End Function

'=====
'Function/Sub Name: Disconnect()
'Description: Disconnect, only if connected
'
'Input:
'    - ServerFileAndPathIn    - Name of FTP server
'    - DestinationFileAndPathIn - Path to save file to
'    - TransferTypeIn         - Binary or Ascii
'    - cbk As cFTPCBK         - For use with call back
'
'Output: Success or failure
'
'References: None
'=====
Sub StartGetFTP(ByVal ServerFileAndPathIn As String, _
    ByVal DestinationFileAndPathIn As String, _
    Optional TransferTypeIn As FileTransferType = ftAscii,
    Optional cbk As cFTPCBK)

    ServerFileAndPath = ServerFileAndPathIn
    DestinationFileAndPath = DestinationFileAndPathIn
    TransferType = TransferTypeIn

    Set SaveCBK = cbk
    'activate the timer that will restart this thread
    Set frmTimer = New frmTimer
    With frmTimer
        Set .Owner = Me
        .Timer1.Interval = 100
        .Timer1.Enabled = True
    End With

End Sub

'=====
'Function/Sub Name: GetFile()
'Description: Get the specified file to the desired location
using
'the specified file transfer type
'
'Input: None
'
'Output: Success or failure
'
'References: None
'=====

```

```

Public Function GetFile() As Boolean

    ' this code is executed when the timer fires for the first
time
    ' unload the form and destroy it completely
    Unload frmTimer
    Set frmTimer = Nothing

    Dim bRet As Boolean
    Dim sFileRemote As String
    Dim sDirRemote As String
    Dim sFileLocal As String
    Dim sTemp As String
    Dim lPos As Long
    Dim sError As String

On Error GoTo vbErrorHandler
'
' If not connected, raise an error
'
    If mlConnection = 0 Then
        On Error GoTo 0
        Err.Raise errNotConnectedToSite, "CGFTP::GetFile",
ERRNOCONNECTION
    End If

' Get the file

    DoEvents
    bRet = FtpGetFile(mlConnection, ServerFileAndPath,
DestinationFileAndPath, False,
FILE_ATTRIBUTE_NORMAL, TransferType Or
INTERNET_FLAG_RELOAD, 0)
    DoEvents

    If bRet = False Then
        sError = ERRNODOWNLOAD
        sError = Replace(sError, "%s", ServerFileAndPath)
        On Error GoTo 0
        GetFile = False
        Err.Raise errGetFileError, "CGFTP::GetFile", sError
    End If

    GetFile = True

    ' inform the client that the process has been completed
    SaveCBK.Complete True
    ' IMPORTANT: destroy the reference to the client
    ' so that it won't be kept alive forever

ExitSub:
    Exit Function

    Set SaveCBK = Nothing

vbErrorHandler:
    GetFile = False
    SaveCBK.Complete True
    GoTo ExitSub
    'Err.Raise errGetFileError, "cFTP::GetFile",
Err.Description

End Function

'=====
'Function/Sub Name: RemoteChDir()

```

```

'Description: Remote Change Directory Command through
WININET
'
'Input:
'   - sDir   - Directory to change to
'
'Output: Success or failure
'
'References: None
'=====
Private Sub RemoteChDir(ByVal sDir As String)
On Error GoTo vbErrorHandler
'
'Remote Change Directory Command through WININET
'
'   Dim sPathFromRoot As String
'   Dim bRet As Boolean
'   Dim sError As String
'
'Needs standard Unix Convention
'
'   sDir = Replace(sDir, "\", "/")
'
'Check for a connection
'
'   If mlConnection = 0 Then
'       On Error GoTo 0
'       Err.Raise errNotConnectedToSite,
"CGFTP::RemoteChDir", ERRNOCONNECTION
'       Exit Sub
'   End If

'   If Len(sDir) = 0 Then
'       Exit Sub
'   Else
'       sPathFromRoot = sDir
'       If Len(sPathFromRoot) = 0 Then
'           sPathFromRoot = "/"
'       End If
'       bRet = FtpSetCurrentDirectory(mlConnection,
sPathFromRoot)
'
'   If we couldn't change directory - raise an error
'
'       If bRet = False Then
'           sError = ERRCHANGEDIRSTR
'           sError = Replace(sError, "%s", sDir)
'           On Error GoTo 0
'           Err.Raise errNoDirChange,
"CGFTP::ChangeDirectory", sError
'       End If
'   End If

'   Exit Sub

vbErrorHandler:
'   Err.Raise Err.Number, "cFTP::RemoteChDir",
Err.Description

End Sub

'=====
'Function/Sub Name: GetINETErrorMsg()
'
'Description: Provide Error information from WinInet.
'
'Input:

```

```

'   - GetINETErrorMsg - Err Num
'
'Output: Detailed error message.
'
'References: None
'=====
Private Function GetINETErrorMsg(ByVal ErrNum As
Long) As String
'   Dim lError As Long
'   Dim lLen As Long
'   Dim sBuffer As String
'
'Get Extra Info from the WinInet.DLL
'
'   If ErrNum = ERROR_INTERNET_EXTENDED_ERROR
Then
'
'Get Message Size and Number
'
'       InternetGetLastResponseInfo lError, vbNullString, lLen
sBuffer = String$(lLen + 1, vbNullChar)
'
'Get Message
'
'       InternetGetLastResponseInfo lError, sBuffer, lLen
GetINETErrorMsg = vbCrLf & sBuffer
'   End If
End Function

*****
'
'Public Property GET and LET statements follow
'
*****

Public Property Let Host(ByVal sHostName As String)
'
'Set the Host Name - only if not connected
'
'   If mlConnection <> 0 Then
'       Err.Raise errInvalidProperty, "ACNFTP:Host_Let",
ERRALREADYCONNECTED
'   End If
'   msHostAddress = sHostName
End Property

Public Property Get Host() As String
'
'Get Host Name
'
'   Host = msHostAddress
End Property

Public Property Let User(ByVal sUserName As String)
'
'Set the user - only if not connected
'
'   If mlConnection <> 0 Then
'       Err.Raise errInvalidProperty, "CGFTP::User_Let",
ERRALREADYCONNECTED
'   End If
'   msUser = sUserName
End Property

```

```

Public Property Get User() As String
'
' Get the user information
'
    User = msUser
End Property

Public Property Let Password(ByVal sPassword As String)
'
' Set the password - only if not connected
'
    If mlConnection <> 0 Then
        Err.Raise errInvalidProperty, "CGFTP::Password_Let",
        ERRALREADYCONNECTED
    End If
    msPassword = sPassword
End Property

Public Property Get Password() As String
'
' Get the password
'
    Password = msPassword
End Property

Public Property Get Directory() As String
'
' Get the directory
'
    Directory = msDirectory
End Property

```

```

Public Property Let Directory(ByVal sDirectory As String)
' Set the directory - only if connected
'
On Error GoTo vbErrorHandler

    Dim sError As String

    If Not (mlConnection = 0) Then
        RemoteChDir sDirectory
        msDirectory = sDirectory
    Else
        On Error GoTo 0
        Err.Raise errNotConnectedToSite,
        "CGFTP::Directory_Let", ERRNOCONNECTION
    End If

    Exit Property

vbErrorHandler:

    Err.Raise errNoDirChange, "CGFTP::Directory[Let]",
    Err.Description

End Property

Public Property Get Connected() As Boolean
'
' Are we connected to an FTP Server ? T/F
'

    Connected = (mlConnection <> 0)
End Property

```



## **CLASS-cFTPCBK**

Option Explicit

#####

' CLASS DESCRIPTION

#####

'Class Name: cFTPCBK

'

'Author: Pat Flanders & Scott Tufts.

'

'Description: Provides and Interface for callback to the HFACS.DLL

'Has no implementation.

'

'References: None

'

#####

'Provide the errorcode back to HFACS

Sub Complete(ErrCode As Boolean)

'

End Sub

## **FORMCLASS-frmTimer**

Option Explicit

Public Owner As cFTP

```
#####  
' FORM DESCRIPTION  
#####  
'Class Name: frmTimer  
'  
'Author: Pat Flanders & Scott Tufts.  
'  
'Description: Provides a timer to give the callback class  
'time to instantiate.  
'  
'References: None  
'  
#####
```

```
Private Sub Timer1_Timer()  
    ' this procedure is executed only once per each invocation  
    ' disable the timer  
    Timer1.Interval = 0  
    Timer1.Enabled = False  
    ' yield to the companion instance  
    Dim bFTPResult As Boolean  
    bFTPResult = Owner.GetFile()  
End Sub
```

## APPENDIX J. INSTALL CD CODE

### FORMCLASS-FrmMain

```

Option Explicit
'#####
'      FORM DESCRIPTION
'#####
'Class Name: FrmMain.frm
'
'Author: Pat Flanders & Scott Tufts
'
'This class is responsible for autorun of the installation CD
and
'providing the user an inface for the install.
'
'References: No special references required.
'
'#####

*****
'      PROPERTIES
*****

Private Declare Function ShellExecute Lib "shell32.dll"
Alias "ShellExecuteA" (ByVal hWnd As Long, ByVal
lpOperation As String, ByVal lpFile As String, ByVal
lpParameters As String, ByVal lpDirectory As String, ByVal
nShowCmd As Long) As Long

Const SW_MAXIMIZE = 3

Dim FileName As String
Dim WorkDir As String
Dim Error As Integer
Dim ErrorMessage As String

'#####
'      FUNCTIONS
'#####

'=====
'Function/Sub Name: Form_Load()
'
'Description: Sets up the initial menu, determines cd drive
letter,
'and plays a sound.
'
'Input: None
'
'Output: None
'
'References: None
'=====

Private Sub Form_Load()

'Determine directory of the CD drive and store it for later
use.
WorkDir = CurDir$()
If Right$(WorkDir, 1) <> "\" Then
    WorkDir = WorkDir & "\"
End If

    Play a sound.
    On Error GoTo 0
    On Error GoTo noSound
    oleSound.DoVerb (1)

noSound:

    'Change menu color when mouse is over button
    Me.lblInstallMSDE.BackColor = &H8000000D
    Me.lblInstallMSDE.ForeColor = &H8000000E

    Me.lblInstallHFACS.BackColor = &H8000000E
    Me.lblInstallHFACS.ForeColor = &H80000008

    Me.lblWin2K.BackColor = &H8000000E
    Me.lblWin2K.ForeColor = &H80000008
    Me.lblDescription.Caption = "Microsoft SQL Server 2000
is the database engine required for HFACS-ME to function."
    & Chr(13) & Chr(13) & _
    "If Microsoft SQL Server 2000 is already installed on this
machine, skip Step 1 and proceed to Step 2." & Chr(13) & _
    "PREREQUISITES: None."

End Sub

'=====
'Function/Sub Name: lblWin2K_Click()
'
'Description: Opens the Step 3 HTML instruction page.
'
'Input: None
'
'Output: None
'
'References: None
'=====

Private Sub lblWin2K_Click()

    FileName = "Step3.htm"
    Screen.MousePointer = vbHourglass

    On Error GoTo StartError

    Error = ShellExecute(0, "open", FileName, "", WorkDir,
SW_MAXIMIZE)
    Me.WaitFor3
    Screen.MousePointer = vbDefault

Exit Sub

```

```

StartError:
    Screen.MousePointer = vbDefault
    MsgBox Err.Description
    MsgBox Err.Number

End Sub

'=====
'Function/Sub Name: lblInstallHFACS_Click()
'
'Description: Launches the HFACS-ME Installation
program.
'
'Input: None
'
'Output: None
'
'References: None
'=====
Private Sub lblInstallHFACS_Click()

    FileName = "HFACS-ME\setup.exe"
    Screen.MousePointer = vbHourglass

    On Error GoTo StartError

    Error = ShellExecute(0, "open", FileName, "", WorkDir,
SW_MAXIMIZE)
    Me.WaitFor3
    Screen.MousePointer = vbDefault

Exit Sub

StartError:
    Screen.MousePointer = vbDefault
    MsgBox Err.Description
    MsgBox Err.Number

End Sub

'=====
'Function/Sub Name:
'    - lblWin2K_MouseMove
'    - lblInstallHFACS_MouseMove
'    - lblInstallMSDE_MouseMove
'
'Description: The next 3 functions are responsible for
changing
'colors of menu buttons in response to mouse movements.
'
'Input: None
'
'Output: None
'
'References: None
'=====
Private Sub lblWin2K_MouseMove(Button As Integer, Shift
As Integer, X As Single, Y As Single)

    ' Change menu color when mouse is over button
    Me.lblInstallMSDE.BackColor = &H8000000E
    Me.lblInstallMSDE.ForeColor = &H80000008

    Me.lblInstallHFACS.BackColor = &H8000000E
    Me.lblInstallHFACS.ForeColor = &H80000008

    Me.lblWin2K.BackColor = &H8000000E
    Me.lblWin2K.ForeColor = &H80000008

    Me.lblDescription.Caption = "Microsoft SQL Server 2000
is the database engine required for HFACS-ME to function."
    & Chr(13) & Chr(13) & _

```

```

Me.lblWin2K.BackColor = &H8000000D
Me.lblWin2K.ForeColor = &H8000000E

    Me.lblDescription.Caption = "If you are installing
HFACS-ME on a computer running Windows 2000 or
Windows NT, you must manually configure settings to allow
users without 'Administrator' permissions to run it." &
Chr(13) & Chr(13) & "Clicking this button will open a link
to an HTML document with detailed instructions outlining
how to make the necessary changes."

    Me.lblDescription.Refresh
    Me.lblInstallMSDE.Refresh
    Me.lblInstallHFACS.Refresh
    Me.lblWin2K.Refresh

End Sub

Private Sub lblInstallHFACS_MouseMove(Button As
Integer, Shift As Integer, X As Single, Y As Single)

    ' Change menu color when mouse is over button
    Me.lblInstallMSDE.BackColor = &H8000000E
    Me.lblInstallMSDE.ForeColor = &H80000008

    Me.lblInstallHFACS.BackColor = &H8000000D
    Me.lblInstallHFACS.ForeColor = &H8000000E

    Me.lblWin2K.BackColor = &H8000000E
    Me.lblWin2K.ForeColor = &H80000008

    Me.lblDescription.Caption = "Installs the HFACS-ME
database and client application." & Chr(13) & Chr(13) & _
"PREREQUISITES: " & Chr(13) & Chr(13) & " 1) IF
ACCESS 2000 IS NOT INSTALLED ON THIS
COMPUTER. There are NO prerequisites. Since you don't
have Access 2000, this installation program will install a
special runtime version." & _
Chr(13) & Chr(13) & " 2) IF ACCESS 2000 IS
ALREADY INSTALLED ON THIS COMPUTER. The
HFACS-ME program REQUIRES Office Service Release 1
or newer to function properly. Since you already have
Access 2000 installed, you must ensure that Microsoft Office
2000 Service Release 1 (or newer) is also installed."

    Me.lblDescription.Refresh
    Me.lblInstallMSDE.Refresh
    Me.lblInstallHFACS.Refresh
    Me.lblWin2K.Refresh

End Sub

Private Sub lblInstallMSDE_MouseMove(Button As Integer,
Shift As Integer, X As Single, Y As Single)

    ' Change menu color when mouse is over button
    Me.lblInstallMSDE.BackColor = &H8000000D
    Me.lblInstallMSDE.ForeColor = &H8000000E

    Me.lblInstallHFACS.BackColor = &H8000000E
    Me.lblInstallHFACS.ForeColor = &H80000008

    Me.lblWin2K.BackColor = &H8000000E
    Me.lblWin2K.ForeColor = &H80000008

    Me.lblDescription.Caption = "Microsoft SQL Server 2000
is the database engine required for HFACS-ME to function."
    & Chr(13) & Chr(13) & _

```

```
"If Microsoft SQL Server 2000 is already installed on this
machine, skip Step 1 and proceed to Step 2." & Chr(13) &
Chr(13) & _
"PREREQUISITES: None."
```

```
Me.lblDescription.Refresh
Me.lblInstallMSDE.Refresh
Me.lblInstallHFACS.Refresh
Me.lblWin2K.Refresh
```

```
End Sub
```

```
'=====
'Function/Sub Name: lblInstallMSDE_Click()
'
'Description: Launches the MSDE Installation program.
'
'Input: None
'
'Output: None
'
'References: None
'=====
```

```
Private Sub lblInstallMSDE_Click()
```

```
MsgBox "Run: " & WorkDir & "HFACS-ME\setup.exe"
```

```
FileName = "MSDE\setup.exe"
```

```
Screen.MousePointer = vbHourglass
On Error GoTo StartError
```

```
Error = ShellExecute(0, "open", FileName, "", WorkDir,
SW_MAXIMIZE)
Me.waitFor3
Screen.MousePointer = vbDefault
```

```
Exit Sub
```

```
StartError:
```

```
Screen.MousePointer = vbDefault
MsgBox Err.Description
MsgBox Err.Number
```

```
End Sub
```

```
'=====
'Function/Sub Name: waitFor3()
'
'Description: Waits for 3 seconds. For future use. Intended
to
'make the form invisible for 3 seconds after a button is
clicked.
'In this way the user can't accidently click another button
while
'the a program is launching.
'
'Input: None
'
'Output: None
'
'References: None
'=====
```

```
Public Sub waitFor3()
```

```
Screen.MousePointer = vbHourglass
Dim PauseTime
Dim Start
Dim i As Integer
PauseTime = 3
Start = Timer ' Set start time.
Do While Timer < Start + PauseTime
DoEvents ' Yield to other processes.
Loop
Screen.MousePointer = vbDefault
```

```
End Sub
```

THIS PAGE INTENTIONALLY LEFT BLANK

## APPENDIX K. INVESTIGATION MODULE

### CLASS-clFrmWindow

Option Compare Database  
Option Explicit

\*\*\*\*\*  
' Type declarations  
\*\*\*\*\*

Private Type RECT     RECT structure used for API calls.  
    Left As Long  
    Top As Long  
    Right As Long  
    Bottom As Long  
End Type

Private Type POINTAPI   POINTAPI structure used for API  
calls.  
    X As Long  
    Y As Long  
End Type

\*\*\*\*\*  
' Member variables  
\*\*\*\*\*  
Private m\_hWnd As Long     Handle of the window.  
Private m\_rctWindow As RECT   Rectangle describing the  
sides of the last polled location of the window.

\*\*\*\*\*  
' Private error constants for use with RaiseError procedure  
\*\*\*\*\*  
Private Const m\_ERR\_INVALIDHWND = 1  
Private Const m\_ERR\_NOPARENTWINDOW = 2

\*\*\*\*\*  
' API function declarations  
\*\*\*\*\*  
Private Declare Function apiIsWindow Lib "user32" Alias  
"IsWindow" (ByVal hwnd As Long) As Long

Private Declare Function apiMoveWindow Lib "user32"  
Alias "MoveWindow" (ByVal hwnd As Long, ByVal X As  
Long, ByVal Y As Long, \_  
    ByVal nWidth As Long, ByVal nHeight As Long, ByVal  
bRepaint As Long) As Long  
    Moves and resizes a window in the coordinate system of  
its parent window.

Private Declare Function apiGetWindowRect Lib "user32"  
Alias "GetWindowRect" (ByVal hwnd As Long, lpRect As  
RECT) As Long  
    After calling, the lpRect parameter contains the RECT  
structure describing the sides of the window in screen  
coordinates.

Private Declare Function apiScreenToClient Lib "user32"  
Alias "ScreenToClient" (ByVal hwnd As Long, lpPoint As  
POINTAPI) As Long  
    Converts lpPoint from screen coordinates to the  
coordinate system of the specified client window.

Private Declare Function apiGetParent Lib "user32" Alias  
"GetParent" (ByVal hwnd As Long) As Long  
    Returns the handle of the parent window of the specified  
window.

#####  
'                   CLASS DESCRIPTION  
#####  
'Class Name: clFormWindow.bas  
'  
'Author: Pat Flanders & Scott Tufts  
'  
'Description: Moves and resizes a window in the coordinate  
system  
' of its parent window.  
'  
'References: None  
'  
#####

\*\*\*\*\*  
'                   FUNCTIONS  
\*\*\*\*\*

=====

'Function/Sub Name: RaiseError()  
'  
'Description: Raises a user-defined error to the calling  
procedure.  
'  
'Input: None  
'  
'Output: None  
'  
'References: None  
'  
=====

Private Sub RaiseError(ByVal lngErrNumber As Long,  
ByVal strErrDesc As String)  
  
    ERR.Raise vbObjectError + lngErrNumber,  
    "clFormWindow", strErrDesc

End Sub

=====

'Function/Sub Name: UpdateWindowRect()  
'  
'Description: Places the current window rectangle position (in  
'pixels, in coordinate system of parent window) in  
'm\_rctWindow.  
'  
'Input: None  
'  
'Output: None

```

'
'References: None
'=====
Private Sub UpdateWindowRect()

    Dim ptCorner As POINTAPI

    If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
        apiGetWindowRect m_hWnd, m_rctWindow
        'm_rctWindow now holds window coordinates in screen
        coordinates.

        If Not Me.Parent Is Nothing Then
            'If there is a parent window, convert top, left of
            window from screen coordinates to parent window
            coordinates.
            With ptCorner
                .X = m_rctWindow.Left
                .Y = m_rctWindow.Top
            End With

            apiScreenToClient Me.Parent.hwnd, ptCorner

            With m_rctWindow
                .Left = ptCorner.X
                .Top = ptCorner.Y
            End With

            'If there is a parent window, convert bottom, right of
            window from screen coordinates to parent window
            coordinates.
            With ptCorner
                .X = m_rctWindow.Right
                .Y = m_rctWindow.Bottom
            End With

            apiScreenToClient Me.Parent.hwnd, ptCorner

            With m_rctWindow
                .Right = ptCorner.X
                .Bottom = ptCorner.Y
            End With
        End If
    Else
        RaiseError m_ERR_INVALIDHWND, "The window
        handle " & m_hWnd & " is no longer valid."
    End If

End Sub

'=====
' Public read-write properties follow
'=====
Public Property Get hwnd() As Long
'Returns the value the user has specified for the window's
handle.

    If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
        hwnd = m_hWnd
    Else
        RaiseError m_ERR_INVALIDHWND, "The window
        handle " & m_hWnd & " is no longer valid."
    End If

End Property

```

```

Public Property Let hwnd(ByVal lngNewValue As Long)
'Sets the window to use by specifying its handle.
'Only accepts valid window handles.

    If lngNewValue = 0 Or apiIsWindow(lngNewValue) Then
        m_hWnd = lngNewValue
    Else
        RaiseError m_ERR_INVALIDHWND, "The value
        passed to the hWnd property is not a valid window handle."
    End If

End Property

Public Property Get Left() As Long
'Returns the current position (in pixels) of the left edge of the
window in the coordinate system of its parent window.

    If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
        UpdateWindowRect
        Left = m_rctWindow.Left
    Else
        RaiseError m_ERR_INVALIDHWND, "The window
        handle " & m_hWnd & " is no longer valid."
    End If

End Property

Public Property Let Left(ByVal lngNewValue As Long)
'Moves the window such that its left edge falls at the position
indicated
'(measured in pixels, in the coordinate system of its parent
window).

    If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
        UpdateWindowRect
        With m_rctWindow
            apiMoveWindow m_hWnd, lngNewValue, .Top,
            .Right - .Left, .Bottom - .Top, True
        End With
    Else
        RaiseError m_ERR_INVALIDHWND, "The window
        handle " & m_hWnd & " is no longer valid."
    End If

End Property

'=====

Public Property Get Top() As Long
'Returns the current position (in pixels) of the top edge of the
window in the coordinate system of its parent window.

    If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
        UpdateWindowRect
        Top = m_rctWindow.Top
    Else
        RaiseError m_ERR_INVALIDHWND, "The window
        handle " & m_hWnd & " is no longer valid."
    End If

End Property

Public Property Let Top(ByVal lngNewValue As Long)
'Moves the window such that its top edge falls at the position
indicated

```



'(measured in pixels, in the coordinate system of its parent window).

```
If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
    UpdateWindowRect
    With m_rctWindow
        apiMoveWindow m_hWnd, .Left, lngNewValue,
        .Right - .Left, .Bottom - .Top, True
    End With
Else
    RaiseError m_ERR_INVALIDHWND, "The window
    handle " & m_hWnd & " is no longer valid."
End If
```

End Property

'-----

Public Property Get Width() As Long  
'Returns the current width (in pixels) of the window.

```
If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
    UpdateWindowRect
    With m_rctWindow
        Width = .Right - .Left
    End With
Else
    RaiseError m_ERR_INVALIDHWND, "The window
    handle " & m_hWnd & " is no longer valid."
End If
```

End Property

Public Property Let Width(ByVal lngNewValue As Long)  
'Changes the width of the window to the value provided (in pixels).

```
If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
    UpdateWindowRect
    With m_rctWindow
        apiMoveWindow m_hWnd, .Left, .Top,
        lngNewValue, .Bottom - .Top, True
    End With
Else
    RaiseError m_ERR_INVALIDHWND, "The window
    handle " & m_hWnd & " is no longer valid."
End If
```

End Property

'-----

Public Property Get Height() As Long  
'Returns the current height (in pixels) of the window.

```
If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
    UpdateWindowRect
    With m_rctWindow
        Height = .Bottom - .Top
    End With
Else
    RaiseError m_ERR_INVALIDHWND, "The window
    handle " & m_hWnd & " is no longer valid."
End If
```

End Property

Public Property Let Height(ByVal lngNewValue As Long)  
'Changes the height of the window to the value provided (in pixels).

```
If m_hWnd = 0 Or apiIsWindow(m_hWnd) Then
    UpdateWindowRect
    With m_rctWindow
        apiMoveWindow m_hWnd, .Left, .Top, .Right - .Left,
        lngNewValue, True
    End With
Else
    RaiseError m_ERR_INVALIDHWND, "The window
    handle " & m_hWnd & " is no longer valid."
End If
```

End Property

'=====

' Public read-only properties follow

Public Property Get Parent() As clFormWindow  
'Returns the parent window as a clFormWindow object.  
'For forms, this should be the Access MDI window.

```
Dim fwParent As New clFormWindow
Dim lngHWnd As Long
```

```
If m_hWnd = 0 Then
    Set Parent = Nothing
ElseIf apiIsWindow(m_hWnd) Then
    lngHWnd = apiGetParent(m_hWnd)
    fwParent.hwnd = lngHWnd
    Set Parent = fwParent
Else
    RaiseError m_ERR_INVALIDHWND, "The window
    handle " & m_hWnd & " is no longer valid."
End If
```

Set fwParent = Nothing

End Property

## **CLASS-INIFile**

Option Explicit

```
'#####  
'          CLASS DESCRIPTION  
'#####  
'Class Name: INIFile.cls  
'  
'Author: Microsoft Corporation. Modified by Pat Flanders  
'&  
'      Scott Tufts  
'  
'This class creates .ini File objects used to create, delete, set,  
'and get values in a standard format Microsoft .ini file. It  
uses  
'calls to the Windows API for efficiency.  
'  
'References: Windows API  
'  
'NOTE: See function headers for internal component  
references.  
'#####
```

```
'*****  
'          PROPERTIES  
'*****
```

```
'The name of the ini file to read  
##ModelId=3B294CFD03A9  
Private msWbkName As String
```

```
'API Wrapper Code - provided by Microsoft  
##ModelId=3B294CFE0000  
Private Declare Function WritePrivateProfileString Lib  
"kernel32" Alias "WritePrivateProfileStringA" (ByVal  
lpApplicationName As String, ByVal lpKeyName As String,  
ByVal lpString As String, ByVal lpFileName As String) As  
Long
```

```
'##ModelId=3B294CFE00AB  
Private Declare Function GetPrivateProfileString Lib  
"kernel32" Alias "GetPrivateProfileStringA" (ByVal  
lpApplicationName As String, ByVal lpKeyName As Any,  
ByVal lpDefault As String, ByVal lpReturnedString As  
String, ByVal nSize As Long, ByVal lpFileName As String)  
As Long
```

```
'##ModelId=3B294CFE0196  
Private Declare Function GetWindowsDirectory Lib  
"kernel32" Alias "GetWindowsDirectoryA" (ByVal lpBuffer  
As String, ByVal nSize As Long) As Long
```

```
'*****  
'          FUNCTIONS  
'*****
```

```
'=====
```

Function/Sub Name: Init()  
'  
'Description: If an instance of a class is created using the  
pseudo-

'constructors from the Constructors.bas module, this function  
is  
'called to pass initial values, thereby mimicking the behavior  
of  
'a constructor with arguments. Passed in values are all  
required, but  
'the Constructors.New\_INIFile() function automatically sets  
'passed-in values to global variable values if they are left  
'blank.  
'

'Input:  
' sPassedInWorkBookName - Name of the .ini file to  
manipulate  
'

'Output: None  
'

'References:  
' - Constructors.bas  
'=====

```
##ModelId=3B294CFE0213  
Friend Sub Init(sPassedInWorkBookName As String)
```

msWbkName = sPassedInWorkBookName

End Sub

```
'=====
```

Function/Sub Name: WriteToIniFile()  
'

'Description: Write a section, key, and value to an .ini file.  
'

'Input:  
' strSection - Name of a section  
' strKey - Name of a key  
' strValue - Name of a key value  
' strFileName - Name of the file to manipulate  
'

'Output: Success or failure  
'

'References: None  
'=====

```
##ModelId=3B294CFE0251  
Friend Function WriteToIniFile(strSection As String, strKey  
As String, strValue As String, strFileName As String) As  
Boolean
```

' Pass in name of section, key, key value, and file name.  
If WritePrivateProfileString(strSection, strKey, \_  
strValue, strFileName) Then  
WriteToIniFile = True  
Else  
MsgBox "Error writing to .ini file: " & Err.LastDllError  
WriteToIniFile = False  
End If

End Function

```
'=====
```

Function/Sub Name: DeleteIniSection()  
'

'Description: Delete a section and all of its keys from an .ini  
file.  
'

```

Input:
' strSection - Name of a section
' strFileName - Name of the file to manipulate
'
Output: Success or failure
References: None
=====
##ModelId=3B294CFE02DE
Friend Function DeleteIniSection(strSection As String,
strFileName As String) As Boolean

    If WritePrivateProfileString(strSection, vbNullString, _
        vbNullString, strFileName) Then
        DeleteIniSection = True
    Else
        MsgBox "Error deleting section from .ini file: " _
            & Err.LastDllError
        DeleteIniSection = False
    End If
End Function

```

```

=====
Function/Sub Name: DeleteIniKey()
Description: Delete a key and its value from an .ini file.
Input:
' strSection - Name of a section
' strKey - Name of a key
' strFileName - Name of the file to manipulate
Output: Success or failure
References: None
=====
##ModelId=3B294CFE033C
Friend Function DeleteIniKey(strSection As String, strKey
As String, strFileName As String) As Boolean

    If WritePrivateProfileString(strSection, strKey, _
        vbNullString, strFileName) Then
        DeleteIniKey = True
    Else
        MsgBox "Error deleting section from .ini file: " _
            & Err.LastDllError
        DeleteIniKey = False
    End If
End Function

```

```

=====
Function/Sub Name: GetIniFileName()
Description: Return name for .ini file. Name includes name
of
'workbook file and ".ini". File path can be made the Windows
directory.
'by uncommenting the code below
Input: None
Output: String path (e.g. C:\windows\HFACS.ini).
References: None

```

```

=====
##ModelId=3B294CFE03A9
Friend Function GetIniFileName() As String

    Dim strWinDir As String
    Dim lngLen As Long

    ' Create null-terminated string to pass to
    ' GetWindowsDirectory.
    strWinDir = String$(255, vbNullChar)

    lngLen = Len(strWinDir)

    ' Return Windows directory.
    GetWindowsDirectory strWinDir, lngLen

    ' Truncate before first null character.
    strWinDir = Left(strWinDir, _
        InStr(strWinDir, vbNullChar) - 1)

    ' Return .ini file name.
    GetIniFileName = strWinDir & "\" & msWbkName &
    ".ini"

```

```

GetIniFileName = App.Path & "\" & msWbkName & ".ini"
End Function

```

```

=====
Function/Sub Name: ReadFromIniFile()
Description: Read a value from an .ini file, given the file
name,
section, key, and default value to return if key is not found.
Input:
' strSection - Name of a section
' strKey - Name of a key
' strDefault - Default name of a key value
' strFileName - Name of the file to manipulate
Output: Success or failure
References: None
=====
##ModelId=3B294CFE03D8
Friend Function ReadFromIniFile(strFileName As String,
strSection As String, strKey As String, Optional strDefault
As String = "") As String

```

```

    Dim strValue As String

    ' Fill string buffer with null characters.
    strValue = String$(255, vbNullChar)

    ' Attempt to read value. GetPrivateProfileString
    ' function returns number of characters written
    ' into string.
    If GetPrivateProfileString(strSection, strKey, _
        strDefault, strValue, Len(strValue), _
        strFileName) > 0 Then
        ' If characters have been written into string, parse string
        ' and return.
        strValue = Left(strValue, InStr(strValue, vbNullChar) -
1)
        ReadFromIniFile = strValue
    Else

```

```
' Otherwise, return a zero-length string.  
ReadFromIniFile = strDefault  
End If  
End Function
```

## **FORMCLASS-1-0-0-0-frm-SelectMishap**

```
Option Compare Database
Option Explicit
#####
'          FORM DESCRIPTION
#####
'Class Name: 1-0-0-0-frm-SelectMishap
'
'Author: Pat Flanders & Scott Tufts
'
'This class displays all the Mishaps in the database and
allows the
'user to sort them by various fields in order to select a mishap
to view or edit. It has buttons that allow initiation of a new
'Mishap or deletion of an existing mishap.
'
'References:
'   - 1-0-0-1-subFrm-SelectMishap
'   - clFormWindow
'   - ez_SizingFunctions
'   - GlobalDeclarations
'
#####

*****
'          FUNCTIONS
*****

=====
'Function/Sub Name: cmdDone_Click()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub cmdDone_Click()

    DoCmd.Quit

End Sub

=====
'Function/Sub Name: cmdViewMishap_Click()
'
'Description: Opens the mishap selected in the subform.
'
'Input: None
'
'Output: None
'
'References: GlobalDeclarations.gLngMishapToGet is a
global variable
holding the value of the mishap ID
'
=====
Private Sub cmdViewMishap_Click()

    On Error GoTo errorHandler
    GlobalDeclarations.gLngMishapToGet =
Me.Manage_Mishaps.Form![MishapID]
    Me.TxtGlobalFocus.Value =
GlobalDeclarations.gLngMishapToGet
    Me.Visible = False

    Dim stLinkCriteria As String
    stLinkCriteria = "[MishapID]= " &
GlobalDeclarations.gLngMishapToGet
    DoCmd.OpenForm "1-0-0-2-frm-EditMishap", , ,
stLinkCriteria
    Exit Sub

errorHandler:
    DoCmd.Beep
    MsgBox "There are no Mishaps to select!", vbOKOnly +
vbExclamation, "Error"

End Sub

=====
'Function/Sub Name: cmdAdd_Click()
'
'Description: Opens the add mishap wizard.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub cmdAdd_Click()

    Me.Visible = False
    DoCmd.OpenForm "1-0-0-5-frm-AddMishap"

End Sub

=====
'Function/Sub Name: cmdKill_Click()
'
'Description: Deletes the mishap selected in the subform.
'
'Input: None
'
'Output: None
'
'References: GlobalDeclarations.gLngMishapToGet is a
global variable
holding the value of the mishap ID
'
=====
Private Sub cmdKill_Click()

    On Error GoTo errorHandler
    GlobalDeclarations.gLngMishapToGet =
Me.Manage_Mishaps.Form![MishapID]
    Me.TxtGlobalFocus.Value =
GlobalDeclarations.gLngMishapToGet
```

```

Dim response As Variant

DoCmd.Beep
response = MsgBox("You are about to permanently delete
the record for MISHAP #" & Me.TxtGlobalFocus.Value & "
and all its related Factors." & Chr(13) & Chr(13) & "It is
STRONGLY recommended that you do not delete mishaps
from the database because this removes all references of
them." & Chr(13) & Chr(13) & "Do you want to delete this
Mishap record despite this warning?", vbYesNo +
vbQuestion + vbDefaultButton2, "Permanently Delete
Mishap?")

If response = vbYes Then

    DoCmd.SetWarnings False
    DoCmd.OpenQuery "1-0-0-2-DeleteMishapAndFactors"
    DoCmd.SetWarnings True
    Me.Manage_Mishaps.Requery

End If

Exit Sub

ErrorHandler:

    DoCmd.Beep
    MsgBox "There are no Mishaps to delete!", vbOKOnly +
vbExclamation, "Error"

End Sub

'=====
'Function/Sub Name: Form_Activate()
'
'Description: Update the menu bar and see if the subform
needs to
'be refreshed.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Activate()

    'Refresh the form if returning from a process that made it
dirty.
    If GlobalDeclarations.gFormNeedsRefresh = True Then
        Me.Manage_Mishaps.Requery
        GlobalDeclarations.gFormNeedsRefresh = False
    End If

End Sub

'=====
'Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'

```

```

'Input: None
'
'Output: None
'
'References:
'    - ezSizeForm
'
'=====
Private Sub Form_Load()
    ezSizeForm Me, -1
    MoveToCenter "1-0-0-0-frm-SelectMishap"
End Sub

'=====
'Function/Sub Name: Form_Open()
'
'Description: Updates the menu bar and sets the MainMenu
form to
'invisible so that the screen is easier to view.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Open(Cancel As Integer)

    'On Error Resume Next

    GlobalDeclarations.synchFileDBTypeToDbValue

    Me.TxtGlobalFocus.Value =
GlobalDeclarations.gLngMishapToGet

    DoCmd.GoToControl "Manage_Mishaps"

End Sub

'=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built -in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'    - clFormWindow
'
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm

```

.hwnd = Forms(strFormName).hwnd	End With
'Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *	Set fwForm = Nothing
0.6)	
.Left = (.Parent.Width - .Width) / 2	End Sub

## **FORMCLASS-1-0-0-1-subfrm-SelectMishap**

```
Option Compare Database
Option Explicit
#####
'          FORM DESCRIPTION
#####
'Class Name: 1-0-0-1-subfrm-SelectMishap
'
'Author: Pat Flanders & Scott Tufts
'
'This class is used in a form/subform relationship with the
'1-0-0-0-firm-SelectMishap form. It displays the mishaps in a
'sortable order.
'
'References:
'   - clFormWindow
'   - ez_SizingFunctions
'   - GlobalDeclarations
'
#####

*****
'          FUNCTIONS
*****

=====
'Function/Sub Name: Form_Open()
'
'Description: Sets color values for the columns in the form as
'well
'as initial sort order.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub Form_Open(Cancel As Integer)

    Me.tglDecending.Value = 0
    Me.OrderBy = "[MishapDate] ASC"
    Me.MishapDate.ForeColor = RGB(10, 140, 50)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
    Me.Type_FK.ForeColor = RGB(0, 0, 0)
    Me.MishapID.ForeColor = RGB(0, 0, 0)

End Sub

=====
'Function/Sub Name: Frame97_AfterUpdate()
'
'Description: Logic module that reacts to radio button clicks.
'Sorts
'the data on the form in the order specified.
'
'Input: None
'
'Output: None
'
'References: None
'
=====

Private Sub Frame97_AfterUpdate()

    If Me.Frame97 = 1 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[MishapDate] DESC"
        Else
            Me.OrderBy = "[MishapDate] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(10, 140, 50)
        Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
        Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If

    If Me.Frame97 = 2 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[OrgID_FK] DESC"
        Else
            Me.OrderBy = "[OrgID_FK] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(0, 0, 0)
        Me.OrgID_FK.ForeColor = RGB(10, 140, 50)
        Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If

    If Me.Frame97 = 3 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[Aircraft_FK] DESC"
        Else
            Me.OrderBy = "[Aircraft_FK] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(0, 0, 0)
        Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
        Me.Aircraft_FK.ForeColor = RGB(10, 140, 50)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If

    If Me.Frame97 = 4 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[Class_FK] DESC"
        Else
            Me.OrderBy = "[Class_FK] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(0, 0, 0)
        Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
        Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
        Me.Class_FK.ForeColor = RGB(10, 140, 50)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If

    If Me.Frame97 = 5 Then
```



```

If Me.tglDecending.Value = -1 Then
    Me.OrderBy = "[MishapLocation] DESC"
Else
    Me.OrderBy = "[MishapLocation] ASC"
End If
Me.MishapDate.ForeColor = RGB(0, 0, 0)
Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
Me.Class_FK.ForeColor = RGB(0, 0, 0)
Me.LocationID_FK.ForeColor = RGB(10, 140, 50)
Me.Type_FK.ForeColor = RGB(0, 0, 0)
Me.MishapID.ForeColor = RGB(0, 0, 0)
End If
If Me.Frame97 = 6 Then
    If Me.tglDecending.Value = -1 Then
        Me.OrderBy = "[Type_FK] DESC"
    Else
        Me.OrderBy = "[Type_FK] ASC"
    End If
    Me.MishapDate.ForeColor = RGB(0, 0, 0)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
    Me.Type_FK.ForeColor = RGB(10, 140, 50)
    Me.MishapID.ForeColor = RGB(0, 0, 0)
End If
If Me.Frame97 = 7 Then
    If Me.tglDecending.Value = -1 Then
        Me.OrderBy = "[MishapID] DESC"
    Else
        Me.OrderBy = "[MishapID] ASC"
    End If
    Me.MishapDate.ForeColor = RGB(0, 0, 0)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
    Me.Type_FK.ForeColor = RGB(0, 0, 0)
    Me.MishapID.ForeColor = RGB(10, 140, 50)
End If

End Sub

'=====
'Function/Sub Name: lblMore_Click()
'
'Description: Reacts to the click of the "More..." box in each
row
'of the data in the form. Opens a form that displays a more
detailed
'description of the mishap because these descriptions are too
big
'to fit in the datagrid of the form.
'
'Input: None
'
'Output: None
'
'References:
'    - 1-0-0-3-PopUpFrm-MishapDescription
'
'=====
Private Sub lblMore_Click()
    GlobalDeclarations.gStrDescription =
Me.lblDescription.Value

```

```

DoCmd.OpenForm "1-0-0-3-PopUpFrm-
MishapDescription"
End Sub

'=====
'Function/Sub Name: tglDecending_AfterUpdate()
'
'Description: Logic module that sorts the data on the form in
'acending or descending order based on the state of the toggle
button.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub tglDecending_AfterUpdate()

    If Me.Frame97 = 1 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[MishapDate] DESC"
        Else
            Me.OrderBy = "[MishapDate] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(10, 140, 50)
        Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
        Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If
    If Me.Frame97 = 2 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[OrgID_FK] DESC"
        Else
            Me.OrderBy = "[OrgID_FK] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(0, 0, 0)
        Me.OrgID_FK.ForeColor = RGB(10, 140, 50)
        Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If
    If Me.Frame97 = 3 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[Aircraft_FK] DESC"
        Else
            Me.OrderBy = "[Aircraft_FK] ASC"
        End If
        Me.MishapDate.ForeColor = RGB(0, 0, 0)
        Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
        Me.Aircraft_FK.ForeColor = RGB(10, 140, 50)
        Me.Class_FK.ForeColor = RGB(0, 0, 0)
        Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
        Me.Type_FK.ForeColor = RGB(0, 0, 0)
        Me.MishapID.ForeColor = RGB(0, 0, 0)
    End If
    If Me.Frame97 = 4 Then
        If Me.tglDecending.Value = -1 Then
            Me.OrderBy = "[Class_FK] DESC"
        Else
            Me.OrderBy = "[Class_FK] ASC"
        End If
    End If

```

```

End If
Me.MishapDate.ForeColor = RGB(0, 0, 0)
Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
Me.Class_FK.ForeColor = RGB(10, 140, 50)
Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
Me.Type_FK.ForeColor = RGB(0, 0, 0)
Me.MishapID.ForeColor = RGB(0, 0, 0)
End If
If Me.Frame97 = 5 Then
    If Me.tglDecending.Value = -1 Then
        Me.OrderBy = "[MishapLocation] DESC"
    Else
        Me.OrderBy = "[MishapLocation] ASC"
    End If
    Me.MishapDate.ForeColor = RGB(0, 0, 0)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(10, 140, 50)
    Me.Type_FK.ForeColor = RGB(0, 0, 0)
    Me.MishapID.ForeColor = RGB(0, 0, 0)
End If
If Me.Frame97 = 6 Then
    If Me.tglDecending.Value = -1 Then
        Me.OrderBy = "[Type_FK] DESC"
    Else

```

```

        Me.OrderBy = "[Type_FK] ASC"
    End If
    Me.MishapDate.ForeColor = RGB(0, 0, 0)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
    Me.Type_FK.ForeColor = RGB(10, 140, 50)
    Me.MishapID.ForeColor = RGB(0, 0, 0)
End If
If Me.Frame97 = 7 Then
    If Me.tglDecending.Value = -1 Then
        Me.OrderBy = "[MishapID] DESC"
    Else
        Me.OrderBy = "[MishapID] ASC"
    End If
    Me.MishapDate.ForeColor = RGB(0, 0, 0)
    Me.OrgID_FK.ForeColor = RGB(0, 0, 0)
    Me.Aircraft_FK.ForeColor = RGB(0, 0, 0)
    Me.Class_FK.ForeColor = RGB(0, 0, 0)
    Me.LocationID_FK.ForeColor = RGB(0, 0, 0)
    Me.Type_FK.ForeColor = RGB(0, 0, 0)
    Me.MishapID.ForeColor = RGB(10, 140, 50)
End If
End Sub

```

## **FORMCLASS-1-0-0-2-frm-EditMishap**

```
Option Compare Database
Option Explicit
#####
'          FORM DESCRIPTION
#####
'Class Name: 1-0-0-2-frm-EditMishap
'
'Author: Pat Flanders & Scott Tufts
'
'This class is used to edit mishaps and add factors. It is
similar
'to the 2-0-1-2-subFrm-View mishaps class, but offers the
additional
'capability to edit the data in the underlying tables.
'
'References:
'   - 1-0-0-7-PopUpFrm-CodeMaintenance
'   - 1-0-0-4-subFrm-Factors
'   - clFormWindow
'   - ez_SizingFunctions
'   - GlobalDeclarations
'
#####
```

```
*****
'          FUNCTIONS
*****
```

```
=====
'Function/Sub Name: cmdCancel_Click()
'
'Description: Closes the form undoing changes BUT ONLY
for events
'that have not already been refreshed. For example, if you
add
'a factor, the entire form is refreshed . . . so clicking cancel
'cannot undo the addition of the factor - you have to use the
'delete button. This function is only capable of undoing
actions
'made to controls in the top portion of the form, and then,
only
'if a refresh has not yet been committed.
'
'Input: None
'
'Output: None
'
'References: None
'
```

```
=====
Private Sub cmdCancel_Click()

On Error GoTo Err_cmdCancel_Click

    DoCmd.DoMenuItem acFormBar, acEditMenu, acUndo, ,
acMenuVer70
    DoCmd.Close

Exit_cmdCancel_Click:
Exit Sub
```

```
Err_cmdCancel_Click:
DoCmd.Close
```

```
End Sub
```

```
=====
'Function/Sub Name: cmdCodeMaintenance_Click()
'
'Description: Opens the code maintenance form.
'
'Input: None
'
'Output: None
'
'References:
'   - 1-0-0-7-PopUpFrm-CodeMaintenance
'
```

```
=====
Private Sub cmdCodeMaintenance_Click()
DoCmd.OpenForm "1-0-0-7-PopUpFrm-
CodeMaintenance"
End Sub
```

```
=====
'Function/Sub Name: cmdSave_Click()
'
'Description: Saves the state of the data and closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
```

```
=====
Private Sub cmdSave_Click()

On Error GoTo Err_Blanks:

DoCmd.Requery
DoCmd.Close
Exit Sub
```

```
Err_Blanks:
DoCmd.Beep
MsgBox "The MishapDate field is a mandatory entry.",
vbOKOnly, "Error"
```

```
End Sub
```

```
=====
'Function/Sub Name: Form_Close()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
```

```

Private Sub Form_Close()
    Forms![1-0-0-0-fm-SelectMishap].Visible = True
End Sub

'=====
'Function/Sub Name: Form_Dirty()
'
'Description: If changes are made to the mishap displayed in
this form
'then the 1-0-0-0-fm-SelectMishap form will need to be
updated when
'this form is closed. This function flags a global variable so
that
'when the 1-0-0-0-fm-SelectMishap form is reactivated, it
refreshes
'to display the changes.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub Form_Dirty(Cancel As Integer)
    MsgBox "The form is now dirty"
    GlobalDeclarations.gFormNeedsRefresh = True
End Sub

'=====
'Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'
'References:
'    - ezSizeForm
'
'=====
Private Sub Form_Load()
    ezSizeForm Me, -1
    MoveToCenter "1-0-0-2-fm-EditMishap"
End Sub

'=====
'Function/Sub Name: Form_Open()
'
'Description: If this form is opened from the 1-0-0-5-fm-
AddMishap
'then the record that was just added needs to be viewed in this
form
'otherwise, it will display the record passed to it in the
'GlobalDeclarations.gLngMishapToGet global variable.
'Input: None
'
'Output: None
'
'References:
'    - GlobalDeclarations
'

```

```

'=====
Private Sub Form_Open(Cancel As Integer)

    'Check to see if you are coming here from the Add Mishap
Wizard or just
    'from the select mishap form.
    If GlobalDeclarations.gBlnAddAMishap = True Then
        'Came from the add form, so close it.
        DoCmd.Close acForm, "1-0-0-5-fm-AddMishap"
        GlobalDeclarations.gBlnAddAMishap = False

        'Set the Title in the form header
        Me.txtTitle.Value = [MishapID] & " - " & [OrgName] &
" - " & [Aircraft_FK]
    Else
        'Set the Title in the form header
        Me.txtTitle.Value = [MishapID] & " - " & [OrgName] &
" - " & [Aircraft_FK]
    End If
End Sub

'=====
'Function/Sub Name: cmdPreview_Click()
'
'Description: Opens the Mishap Snapshot report.
'
'Input: None
'
'Output: None
'
'References:
'    - 1-0-MishapSnapshot-OpenMishaps
'
'=====
Private Sub cmdPreview_Click()

    Me.Refresh

    GlobalDeclarations.gLngMishapToGet = Me.txtMishapID

    On Error GoTo StartError
    Dim stDocName As String
    Dim stLinkCriteria As String
    stDocName = "1-0-MishapSnapshot-OpenMishaps"
    stLinkCriteria = "[MishapID]=" &
GlobalDeclarations.gLngMishapToGet
    DoCmd.OpenReport stDocName, A_PREVIEW, ,
stLinkCriteria

    Exit Sub

StartError:

    DoCmd.Beep
    MsgBox "There are no Mishaps to select or you do not have
a default printer installed.", vbOKOnly + vbExclamation,
"Error"

End Sub

'=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm

```

```
'class breaks Access's built -in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'      - clFormWindow
'
```

```
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        '.Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

End Sub
```

### **FORMCLASS-1-0-0-3-PopUpFrm-MishapDescription**

```
Option Compare Database
#####
'          FORM DESCRIPTION
#####
'Class Name: 1-0-0-3-PopUpFrm-MishapDescription
'
'Author: Pat Flanders & Scott Tufts
'
'This class is
'
'References:
'   - clFormWindow
'   - ez_SizingFunctions
'   - GlobalDeclarations
'
#####

*****
'          FUNCTIONS
*****

'=====
Function/Sub Name: cmdDone_Click()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====
Private Sub cmdDone_Click()
    DoCmd.Close acForm, "1-0-0-3-PopUpFrm-MishapDescription"
End Sub

'=====
Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'
'References:
'   - ezSizeForm
'
'=====
Private Sub Form_Load()
    ezSizeForm Me, -1
    MoveToCenter "1-0-0-3-PopUpFrm-MishapDescription"
```

```
End Sub

'=====
Function/Sub Name: Form_Open()
'
'Description: Updates the menu bar and sets shows the value
of the
'description for the mishap stored in the
GlobalDeclarations.gStrDescription
'global variable.
'
'Input: None
'
'Output: None
'
'References:
'   - GlobalDeclarations
'
'=====
Private Sub Form_Open(Cancel As Integer)

    Me.txtDescription = GlobalDeclarations.gStrDescription
End Sub

'=====
Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
'   - clFormWindow
'
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        .Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing
End Sub
```

## **FORMCLASS-1-0-0-4-Subfrm-Factors**

```
'Option Compare Database
Option Explicit
'#####
'          FORM DESCRIPTION
'#####
'Class Name: 1-0-0-4-subfrm-Factors
'
'Author: Pat Flanders & Scott Tufts
'
'This class is used in a form/subform relationship with the
'1-0-0-2-frm-EditMishap form to display, add, and delete
factors
'to a mishap.
'
References:
'   - 1-0-0-2-frm-EditMishap
'   - clFormWindow
'   - ez_SizingFunctions
'   - GlobalDeclarations
'#####

*****
'          FUNCTIONS
*****
*****

=====
Function/Sub Name: cmdAddFactor_Click()
'
Description: Adds a blank factor to the mishap indicated by
the
GlobalDeclarations.gLngMishapToGet global variable.
'
Input: None
'
Output: None
'
References:
'   - GlobalDeclarations
'=====
Private Sub cmdAddFactor_Click()

    On Error GoTo Err_cmdAddFactor_Click

    DoCmd.SetWarnings (False) 'Turn off warning messages
    Me.AllowAdditions = True 'Toggle the form to allow
addition of records

    DoCmd.GoToRecord , , acNewRec 'Create a new record

    Me.txtMishapID.Value =
GlobalDeclarations.gLngMishapToGet 'Set the value of the
Mishap
    Me.txtFactorSummary.Value = "Please enter a short
summary description of the Factor."
    Me.cbo3rdLevelCode.Value = "UNK"
    DoCmd.DoMenuItem acFormBar, acRecordsMenu,
acSaveRecord, , acMenuVer70 'Save the record
    Me.AllowAdditions = False 'Toggle back to not allow
addition of records
    Me.Refresh 'Refresh so the user can see the changes
    Me.Recordset.MoveLast 'Move to the record just created
    DoCmd.SetWarnings (True)

Exit_cmdAddFactor_Click:
Exit Sub

Err_cmdAddFactor_Click:

    MsgBox ERR.Description
    Resume Exit_cmdAddFactor_Click

End Sub

=====
Function/Sub Name: cmdDelFactor_Click()
'
Description: Deletes the factor with the current focus.
'
Input: None
'
Output: None
'
References: None
'=====
Private Sub cmdDelFactor_Click()
On Error GoTo Err_cmdDelFactor_Click

    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, ,
acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, ,
acMenuVer70

Exit_cmdDelFactor_Click:
Exit Sub

Err_cmdDelFactor_Click:
    MsgBox ERR.Description
    Resume Exit_cmdDelFactor_Click

End Sub
```

## **FORMCLASS-1-0-0-5-frm-AddMishap**

Option Compare Database  
Option Explicit

'Placekeeper for current wizard page number.  
Dim iPageNumber As Integer

'Tracks position of 1st Level Factor being input.  
Dim iFirstLevelCounter As Integer

'For hiding the back button when appropriate  
Dim bHideBackButton As Boolean

'Tracks number of factors added so far  
Dim iFactorsAddedCounter As Integer

'For closing the program and returning to main.  
Dim bBackToMain As Boolean

#####  
' FORM DESCRIPTION  
#####

'Class Name: 1-0-0-5-frm-AddMishap

'Author: Pat Flanders & Scott Tufts

'This class is a wizard used to add Mishaps to the database.  
The  
'illusion of many forms is created using a TAB control on the  
form

'and setting the "tab style" property to "None". THIS IS  
IMPORTANT.

'The only way to edit the other pages of the tab control is to  
'set the tab property to "Tabs" when the form is in design  
view

'and then change it back to "None" when finished. If you  
don't

'do this, you cannot edit any of the pages of the wizard except  
'the first one.

'After a mishap is added, the 1-0-0-2-frm-EditMishap form is  
'opened with the newly added Mishap selected for editing.  
This

'allows the user to immediately add Factors without having to  
'go back to the main menu.

'References:

' - 1-0-0-7-PopUpFrm-CodeMaintenance  
' - 1-0-0-2-frm-EditMishap  
' - clFormWindow  
' - ez\_SizingFunctions  
' - GlobalDeclarations

#####

\*\*\*\*\*  
' FUNCTIONS  
\*\*\*\*\*

=====

'Function/Sub Name: cmdBack\_Click()

'  
'Description: Switches form focus back one tab in the tab  
view

'control.

'Input: None

'Output: None

'References: None

=====

Private Sub cmdBack\_Click()

businessLogicBackward (iPageNumber)

End Sub

=====

'Function/Sub Name: cmdNext\_Click()

'Description: Switches form focus forward one tab in the tab  
view

'control.

'Input: None

'Output: None

'References: None

=====

Private Sub cmdNext\_Click()

If iPageNumber = 0 Then iPageNumber = iPageNumber +  
1  
businessLogicForward (iPageNumber)

End Sub

=====

'Function/Sub Name: cmdFinish\_Click()

'Description: Adds the mishap to the database and opens the  
edit  
'form so that the user can add factors.

'Input: None

'Output: None

'References:

' - 1-0-0-2-frm-EditMishap

=====

Private Sub cmdFinish\_Click()

On Error GoTo StartError

Me.Visible = False

Dim stLinkCriteria As String

stLinkCriteria = "[MishapID]= " &

GlobalDeclarations.gLngMishapToGet



```

    DoCmd.OpenForm "1-0-0-2-fm-EditMishap", , ,
stLinkCriteria

ExitSub:
    Exit Sub

StartError:
    DoCmd.Beep
    MsgBox "You have left at least one field in this wizard
blank. All entries are mandatory. Please go back and input
data for all fields.", vbOKOnly, "All Entries Are Mandatory"
    Resume ExitSub

End Sub

```

```

=====
'Function/Sub Name: cmdCodeMaintenance_Click()
'
'Description: Opens the code maintenance form.
'
'Input: None
'Output: None
'
'References:
'    - 1-0-0-7-PopUpFrm-CodeMaintenance
'
=====
Private Sub cmdCodeMaintenance_Click()
    DoCmd.OpenForm "1-0-0-7-PopUpFrm-
CodeMaintenance"
End Sub

```

```

=====
'Function/Sub Name:
'    - cmdCrewCoord_Click()
'    - cmdEnvironmental_Click()
'    - cmdEquipment_Click()
'    - cmdError_Click()
'    - cmdMedical_Click()
'    - cmdOrganizational_Click()
'    - cmdReadiness_Click()
'    - cmdSupervisory_Click()
'    - cmdViolation_Click()
'    - cmdWorkspace_Click()
'
'Description: For controlling movement between pages not
capable of
'movement using the "next" function
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub cmdCrewCoord_Click()
    iPageNumber = 11
    DoCmd.GoToControl "Page11"
    Me.cmdNext.Enabled = True
End Sub

```

```

Private Sub cmdEnvironmental_Click()
    iPageNumber = 13
    DoCmd.GoToControl "Page13"

```

```

    Me.cmdNext.Enabled = True
End Sub

```

```

Private Sub cmdEquipment_Click()
    iPageNumber = 14
    DoCmd.GoToControl "Page14"
    Me.cmdNext.Enabled = True
End Sub

```

```

Private Sub cmdError_Click()
    iPageNumber = 16
    DoCmd.GoToControl "Page16"
    Me.cmdNext.Enabled = True
End Sub

```

```

Private Sub cmdMedical_Click()
    iPageNumber = 10
    DoCmd.GoToControl "Page10"
    Me.cmdNext.Enabled = True
End Sub

```

```

Private Sub cmdOrganizational_Click()
    iPageNumber = 8
    DoCmd.GoToControl "Page8"
    Me.cmdNext.Enabled = True
End Sub

```

```

Private Sub cmdReadiness_Click()
    iPageNumber = 12
    DoCmd.GoToControl "Page12"
    Me.cmdNext.Enabled = True
End Sub

```

```

Private Sub cmdSupervisory_Click()
    iPageNumber = 9
    DoCmd.GoToControl "Page9"
    Me.cmdNext.Enabled = True
End Sub

```

```

Private Sub cmdViolation_Click()
    iPageNumber = 17
    DoCmd.GoToControl "Page17"
    Me.cmdNext.Enabled = True
End Sub

```

```

Private Sub cmdWorkspace_Click()
    iPageNumber = 15
    DoCmd.GoToControl "Page15"
    Me.cmdNext.Enabled = True
End Sub

```

```

=====
'Function/Sub Name: Form_Close()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References:
'    - 1-0-0-0-fm-SelectMishap
'
=====
Private Sub Form_Close()

```

```

    If bBackToMain = True Then
        Forms![1-0-0-0-fm-SelectMishap].Visible = True
    End If
End Sub

```

```

End If

End Sub

'=====
'Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'
'References:
'    - ezSizeForm
'
'=====

Private Sub Form_Load()
    ezSizeForm Me, -1
    MoveToCenter "1-0-0-5-frm-AddMishap"
End Sub

'=====

'Function/Sub Name: Form_Open(Cancel As Integer)
'
'Description: Initializes all variables.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====

Private Sub Form_Open(Cancel As Integer)

    bBackToMain = False

    'Set initial values on page 1
    Me.txtDate.Value = Format(Now(), "dd-mmm-yyyy")
    Me.cboAircraftType.Value = "Unknown"
    Me.cboOrganization.Value = "UNK"
    Me.cboLocation.Value = "UNK"
    Me.txtShortDescription.Value = "Please enter a short
description."
    Me.txtLongDescription.Value = "Please enter a long
description."

    'Set the database type
    GlobalDeclarations.getDBType
    Me.txtDatabaseType.Value =
GlobalDeclarations.gstrDatabaseType

    'Set initial value of the checkboxes on page 18
    Me.chkP18MgmtCond.Value = False
    Me.chkP18MaintCond.Value = False
    Me.chkP18WorkCond.Value = False
    Me.chkP18MaintActs.Value = False

    'Set initial values of combo and text boxes on pages 8-17
    Me.cbo3rdLevelCode8.Value = "DES"
    Me.cbo3rdLevelCode9.Value = "IDQ"
    Me.cbo3rdLevelCode10.Value = "LIM"

```

```

    Me.cbo3rdLevelCode11.Value = "ADA"
    Me.cbo3rdLevelCode12.Value = "CRT"
    Me.cbo3rdLevelCode13.Value = "EHZ"
    Me.cbo3rdLevelCode14.Value = "DUC"
    Me.cbo3rdLevelCode15.Value = "CON"
    Me.cbo3rdLevelCode16.Value = "JDG"
    Me.cbo3rdLevelCode17.Value = "IFC"
    Me.txtFactorSummary8.Value = "No description entered,
yet."
    Me.txtFactorSummary9.Value = "No description entered,
yet."
    Me.txtFactorSummary10.Value = "No description entered,
yet."
    Me.txtFactorSummary11.Value = "No description entered,
yet."
    Me.txtFactorSummary12.Value = "No description entered,
yet."
    Me.txtFactorSummary13.Value = "No description entered,
yet."
    Me.txtFactorSummary14.Value = "No description entered,
yet."
    Me.txtFactorSummary15.Value = "No description entered,
yet."
    Me.txtFactorSummary16.Value = "No description entered,
yet."
    Me.txtFactorSummary17.Value = "No description entered,
yet."

```

```

'Set the initial value of the factors counter
iFactorsAddedCounter = 0
Me.txtFactorCounter.Value = iFactorsAddedCounter

```

End Sub

```

'=====
'Function/Sub Name: txtDate_GotFocus()
'
'Description: Ensures date fields are properly formatted to
medium
'date.
'
'Input: None
'
'Output: None
'
'References: None
'
'=====

Private Sub txtDate_GotFocus()
    'Format the date in the textbox so the time doesn't appear
    Me.txtDate = Format([txtDate], "Medium Date")
End Sub

```

```

*****
' LOGIC SUBROUTINES
*****

```

```

'=====
'Function/Sub Name: businessLogicForward()
'
'Description: Logic to determine what page to go in the
forward
'direction.
'

```

```

Input:
' - pageCurrentlyAt - The page with the current focus.
'
'Output: None
'
'References: None
'
=====
Private Sub businessLogicForward(pageCurrentlyAt As Integer)

    Select Case pageCurrentlyAt

        Case 1
            If Trim(Me.txtLongDescription.Value) = "" Then
                Me.txtLongDescription.Value = "Please enter a long description."
            End If
            If IsNull(Me.txtLongDescription.Value) Then
                Me.txtLongDescription.Value = "Please enter a long description."
            End If
            Me.cmdBack.Enabled = True
            Me.cmdCodeMaintenance.Visible = False
            iPageNumber = iPageNumber + 1
            DoCmd.GoToControl "Page" & iPageNumber

        Case 2
            Select Case Me.fraInjuries

                Case 1 To 2
                    Me.cboClass.Value = "A"

                Case 3 To 4
                    If Me.fraDamage = 1 Or Me.fraDamage = 2
Then
                        Me.cboClass.Value = "A"
                    Else
                        Me.cboClass.Value = "B"
                    End If

                Case 5
                    If Me.fraDamage = 1 Or Me.fraDamage = 2
Then
                        Me.cboClass.Value = "A"
                    ElseIf Me.fraDamage = 3 Then
                        Me.cboClass.Value = "B"
                    Else
                        Me.cboClass.Value = "C"
                    End If

                Case 6
                    If Me.fraDamage = 1 Or Me.fraDamage = 2
Then
                        Me.cboClass.Value = "A"
                    ElseIf Me.fraDamage = 3 Then
                        Me.cboClass.Value = "B"
                    ElseIf Me.fraDamage = 4 Then
                        Me.cboClass.Value = "C"
                    ElseIf Me.fraDamage = 5 Then
                        MsgBox "The criteria you selected for damage and injuries " & _
                            "does not qualify as a reportable mishap.",
vbOKOnly + vbInformation, "Mishap Does Not Qualify"
                    Exit Sub
                    End If

            End Select

    End Select

```

```

iPageNumber = iPageNumber + 1
DoCmd.GoToControl "Page" & iPageNumber

Case 3

    Select Case Me.fraType

        Case 1
            If Me.fraType = 1 Then
                Me.cboType = "FM"
            End If

        Case 2
            If Me.fraType = 2 Then
                Me.cboType = "FRM"
            End If

        Case 3
            If Me.fraType = 3 Then
                Me.cboType = "AGM"
            End If

    End Select

    'Code to save the mishap goes here
    addMishap
    GlobalDeclarations.gLngMishapToGet =
Me.TxtGlobalFocus.Value
    GlobalDeclarations.gBlnAddAMishap = True
    GlobalDeclarations.gFormNeedsRefresh = True

    Me.cmdBack.Enabled = False
    iPageNumber = 18
    iFirstLevelCounter = 1
    DoCmd.GoToControl "Page18"

Case 4 To 7
    'Do nothing. Button is disabled

Case 8
    If Trim(Me.txtFactorSummary8.Value) = "" Then
        Me.txtFactorSummary8.Value = "No description entered, yet."
    End If

    If Trim(Me.cbo3rdLevelCode8.Value) = "" Then
        MsgBox "You can't leave the 3RD LEVEL FACTOR blank.", vbOKOnly, "Missing Mandatory Entry"
    Else
        addFactor Me.cbo3rdLevelCode8.Value,
Me.txtFactorSummary8.Value
        Me.txtFactorSummary8.Value = "No description entered, yet."
        MsgBox "Factor added to database.", vbOKOnly + vbInformation, "Success"
        If iFirstLevelCounter = 1 Then
            Me.cmdBack.Enabled = False
            iPageNumber = 18
            DoCmd.GoToControl "Page18"
        End If

Case 9
    If Trim(Me.txtFactorSummary9.Value) = "" Then
        Me.txtFactorSummary9.Value = "No description entered, yet."
    End If

```

```

        If Trim(Me.cbo3rdLevelCode9.Value) = "" Then
            MsgBox "You can't leave the 3RD LEVEL
            FACTOR blank.", vbOKOnly, "Missing Mandatory Entry"
        Else
            addFactor Me.cbo3rdLevelCode9.Value,
            Me.txtFactorSummary9.Value
            Me.txtFactorSummary9.Value = "No description
            entered, yet."
            MsgBox "Factor added to database.", vbOKOnly +
            vbInformation, "Success"
            If iFirstLevelCounter = 1 Then
                Me.cmdBack.Enabled = False
                iPageNumber = 18
                DoCmd.GoToControl "Page18"
            End If
        End If
    
```

```

    Case 10
        If Trim(Me.txtFactorSummary10.Value) = "" Then
            Me.txtFactorSummary10.Value = "No description
            entered, yet."
        End If
    
```

```

        If Trim(Me.cbo3rdLevelCode10.Value) = "" Then
            MsgBox "You can't leave the 3RD LEVEL
            FACTOR blank.", vbOKOnly, "Missing Mandatory Entry"
        Else
            addFactor Me.cbo3rdLevelCode10.Value,
            Me.txtFactorSummary10.Value
            Me.txtFactorSummary10.Value = "No description
            entered, yet."
            MsgBox "Factor added to database.", vbOKOnly +
            vbInformation, "Success"
            If iFirstLevelCounter = 1 Then
                Me.cmdBack.Enabled = False
                iPageNumber = 18
                DoCmd.GoToControl "Page18"
            End If
        End If
    
```

```

    Case 11
        If Trim(Me.txtFactorSummary11.Value) = "" Then
            Me.txtFactorSummary11.Value = "No description
            entered, yet."
        End If
    
```

```

        If Trim(Me.cbo3rdLevelCode11.Value) = "" Then
            MsgBox "You can't leave the 3RD LEVEL
            FACTOR blank.", vbOKOnly, "Missing Mandatory Entry"
        Else
            addFactor Me.cbo3rdLevelCode11.Value,
            Me.txtFactorSummary11.Value
            Me.txtFactorSummary11.Value = "No description
            entered, yet."
            MsgBox "Factor added to database.", vbOKOnly +
            vbInformation, "Success"
            If iFirstLevelCounter = 1 Then
                Me.cmdBack.Enabled = False
                iPageNumber = 18
                DoCmd.GoToControl "Page18"
            End If
        End If
    
```

```

    Case 12
        If Trim(Me.txtFactorSummary12.Value) = "" Then
            Me.txtFactorSummary12.Value = "No description
            entered, yet."
        End If
    
```

```

        If Trim(Me.cbo3rdLevelCode12.Value) = "" Then
    
```

```

            MsgBox "You can't leave the 3RD LEVEL
            FACTOR blank.", vbOKOnly, "Missing Mandatory Entry"
        Else
            addFactor Me.cbo3rdLevelCode12.Value,
            Me.txtFactorSummary12.Value
            Me.txtFactorSummary12.Value = "No description
            entered, yet."
            MsgBox "Factor added to database.", vbOKOnly +
            vbInformation, "Success"
            If iFirstLevelCounter = 1 Then
                Me.cmdBack.Enabled = False
                iPageNumber = 18
                DoCmd.GoToControl "Page18"
            End If
        End If
    
```

```

    Case 13
        If Trim(Me.txtFactorSummary13.Value) = "" Then
            Me.txtFactorSummary13.Value = "No description
            entered, yet."
        End If
    
```

```

        If Trim(Me.cbo3rdLevelCode13.Value) = "" Then
            MsgBox "You can't leave the 3RD LEVEL
            FACTOR blank.", vbOKOnly, "Missing Mandatory Entry"
        Else
            addFactor Me.cbo3rdLevelCode13.Value,
            Me.txtFactorSummary13.Value
            Me.txtFactorSummary13.Value = "No description
            entered, yet."
            MsgBox "Factor added to database.", vbOKOnly +
            vbInformation, "Success"
            If iFirstLevelCounter = 1 Then
                Me.cmdBack.Enabled = False
                iPageNumber = 18
                DoCmd.GoToControl "Page18"
            End If
        End If
    
```

```

    Case 14
        If Trim(Me.txtFactorSummary14.Value) = "" Then
            Me.txtFactorSummary14.Value = "No description
            entered, yet."
        End If
    
```

```

        If Trim(Me.cbo3rdLevelCode14.Value) = "" Then
            MsgBox "You can't leave the 3RD LEVEL
            FACTOR blank.", vbOKOnly, "Missing Mandatory Entry"
        Else
            addFactor Me.cbo3rdLevelCode14.Value,
            Me.txtFactorSummary14.Value
            Me.txtFactorSummary14.Value = "No description
            entered, yet."
            MsgBox "Factor added to database.", vbOKOnly +
            vbInformation, "Success"
            If iFirstLevelCounter = 1 Then
                Me.cmdBack.Enabled = False
                iPageNumber = 18
                DoCmd.GoToControl "Page18"
            End If
        End If
    
```

```

    Case 15
        If Trim(Me.txtFactorSummary15.Value) = "" Then
            Me.txtFactorSummary15.Value = "No description
            entered, yet."
        End If
    
```

```

        If Trim(Me.cbo3rdLevelCode15.Value) = "" Then
            MsgBox "You can't leave the 3RD LEVEL
            FACTOR blank.", vbOKOnly, "Missing Mandatory Entry"
        End If
    
```

```

Else
    addFactor Me.cbo3rdLevelCode15.Value,
Me.txtFactorSummary15.Value
    Me.txtFactorSummary15.Value = "No description
entered, yet."
    MsgBox "Factor added to database.", vbOKOnly +
vbInformation, "Success"
    If iFirstLevelCounter = 1 Then
Me.cmdBack.Enabled = False
        iPageNumber = 18
        DoCmd.GoToControl "Page18"
    End If

Case 16
    If Trim(Me.txtFactorSummary16.Value) = "" Then
        Me.txtFactorSummary16.Value = "No description
entered, yet."
    End If

    If Trim(Me.cbo3rdLevelCode16.Value) = "" Then
        MsgBox "You can't leave the 3RD LEVEL
FACTOR blank.", vbOKOnly, "Missing Mandatory Entry"
    Else
        addFactor Me.cbo3rdLevelCode16.Value,
Me.txtFactorSummary16.Value
        Me.txtFactorSummary16.Value = "No description
entered, yet."
        MsgBox "Factor added to database.", vbOKOnly +
vbInformation, "Success"
        If iFirstLevelCounter = 1 Then
Me.cmdBack.Enabled = False
            iPageNumber = 18
            DoCmd.GoToControl "Page18"
        End If

Case 17
    If Trim(Me.txtFactorSummary17.Value) = "" Then
        Me.txtFactorSummary17.Value = "No description
entered, yet."
    End If

    If Trim(Me.cbo3rdLevelCode17.Value) = "" Then
        MsgBox "You can't leave the 3RD LEVEL
FACTOR blank.", vbOKOnly, "Missing Mandatory Entry"
    Else
        addFactor Me.cbo3rdLevelCode17.Value,
Me.txtFactorSummary17.Value
        Me.txtFactorSummary17.Value = "No description
entered, yet."
        MsgBox "Factor added to database.", vbOKOnly +
vbInformation, "Success"
        If iFirstLevelCounter = 1 Then
Me.cmdBack.Enabled = False
            iPageNumber = 18
            DoCmd.GoToControl "Page18"
        End If

Case 18
    askWhereToGo

Case 19
    'Do nothing. Button is disabled

End Select
End Sub

```

```

'=====
'Function/Sub Name: businessLogicBackward()
'
'Description: Logic to determine what page to go in the
Reverse
'direction.
'
'Input:
'    - pageCurrentlyAt - The page with the current focus.
'
'Output: None
'
'References: None
'
'=====
Private Sub businessLogicBackward(pageCurrentlyAt As
Integer)

    Select Case pageCurrentlyAt

        Case 1
            'Do nothing. Back button is disabled

        Case 2
            iPageNumber = iPageNumber - 1
            DoCmd.GoToControl "Page" & iPageNumber
            Me.cmdCodeMaintenance.Visible = True
            Me.cmdBack.Enabled = False

        Case 3
            iPageNumber = iPageNumber - 1
            DoCmd.GoToControl "Page" & iPageNumber

        Case 4 To 7
            iPageNumber = 18
            DoCmd.GoToControl "Page18"
            Me.cmdBack.Enabled = False
            Me.cmdNext.Enabled = True

        Case 8 To 9
            iPageNumber = 4
            DoCmd.GoToControl "Page4"
            Me.cmdNext.Enabled = False

        Case 10 To 12
            iPageNumber = 5
            DoCmd.GoToControl "Page5"
            Me.cmdNext.Enabled = False

        Case 13 To 15
            iPageNumber = 6
            DoCmd.GoToControl "Page6"
            Me.cmdNext.Enabled = False

        Case 16 To 17
            iPageNumber = 7
            DoCmd.GoToControl "Page7"
            Me.cmdNext.Enabled = False

        Case 18
            If iFirstLevelCounter > 1 Then
                iFirstLevelCounter = iFirstLevelCounter - 1

                'Update the page 18 to reflect backwards
movement.
                Select Case iFirstLevelCounter

                    Case 1 'managementCond

```

```

        With Me.lblP18MgmtCond
            .ForeColor = QBColor(9)
            .FontWeight = 600
            .Caption = "Input MANAGMENT
CONDITIONS related factors."
        End With
        Me.chkP18MgmtCond.Value = False
        With Me.lblP18MaintCond
            .ForeColor = QBColor(0)
            .FontWeight = 400
        End With
        DoCmd.GoToControl "cmdNext"
        Me.cmdBack.Enabled = False

    Case 2 'maintainerCond
        With Me.lblP18MaintCond
            .ForeColor = QBColor(9)
            .FontWeight = 600
            .Caption = "Input MAINTAINER
CONDITIONS related factors."
        End With
        Me.chkP18MaintCond.Value = False
        With Me.lblP18WorkCond
            .ForeColor = QBColor(0)
            .FontWeight = 400
        End With

    Case 3 'workingCond
        With Me.lblP18WorkCond
            .ForeColor = QBColor(9)
            .FontWeight = 600
            .Caption = "Input WORKING
CONDITIONS related factors."
        End With
        Me.chkP18WorkCond.Value = False
        With Me.lblP18MaintActs
            .ForeColor = QBColor(0)
            .FontWeight = 400
        End With

    Case 4 'maintainerAct
        With Me.lblP18MaintActs
            .ForeColor = QBColor(9)
            .FontWeight = 600
            .Caption = "Input WORKING
CONDITIONS related factors."
        End With
        Me.chkP18MaintActs.Value = False

    End Select
Else
    MsgBox "The Mishap has already been entered
into the database and cannot be edited from this wizard." & _
    Chr(13) & Chr(13) & "You can edit the mishap
data after you have finished entering factor data.",
vbOKOnly, _
    "Can't Edit Mishap"
End If

Case 19
    iPageNumber = 18
    DoCmd.GoToControl "Page18"
    Me.cmdFinish.Enabled = False

End Select

End Sub

```

```

=====
'Function/Sub Name: askWhereToGo()
'
'Description: Logic to determine what page to go to based on
user
'input.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub askWhereToGo()

startSelect:

    Select Case iFirstLevelCounter

    Case 1 'managementCond
        If managementCond = True Then
            iPageNumber = 4
            DoCmd.GoToControl "Page4"
            Me.cmdNext.Enabled = False
            Me.cmdBack.Enabled = True
        Else
            iFirstLevelCounter = 2
            Me.cmdBack.Enabled = True
            With Me.lblP18MgmtCond
                .ForeColor = QBColor(8)
                .FontWeight = 400
                .Caption = "COMPLETED - Input
MANAGMENT CONDITIONS related factors."
            End With
            Me.chkP18MgmtCond.Value = True
            With Me.lblP18MaintCond
                .ForeColor = QBColor(9)
                .FontWeight = 600
            End With
            iPageNumber = 18
            DoCmd.GoToControl "Page18"
            GoTo startSelect
        End If

    Case 2 'maintainerCond
        If maintainerCond = True Then
            iPageNumber = 5
            DoCmd.GoToControl "Page5"
            Me.cmdNext.Enabled = False
            Me.cmdBack.Enabled = True
        Else
            iFirstLevelCounter = 3
            With Me.lblP18MaintCond
                .ForeColor = QBColor(8)
                .FontWeight = 400
                .Caption = "COMPLETED - Input
MAINTAINER CONDITIONS related factors."
            End With
            Me.chkP18MaintCond.Value = True
            With Me.lblP18WorkCond
                .ForeColor = QBColor(9)
                .FontWeight = 600
            End With
            iPageNumber = 18
            DoCmd.GoToControl "Page18"
            GoTo startSelect
        End If
    End Select
End Sub

```

```

End If

Case 3 'workingCond
If workingCond = True Then
    iPageNumber = 6
    DoCmd.GoToControl "Page6"
    Me.cmdNext.Enabled = False
    Me.cmdBack.Enabled = True
Else
    iFirstLevelCounter = 4
    With Me.lblP18WorkCond
        .ForeColor = QBColor(8)
        .FontWeight = 400
        .Caption = "COMPLETED - Input WORKING
CONDITIONS related factors."
    End With
    Me.chkP18WorkCond.Value = True
    With Me.lblP18MaintActs
        .ForeColor = QBColor(9)
        .FontWeight = 600
    End With
    iPageNumber = 18
    DoCmd.GoToControl "Page18"
    'GoTo startSelect
End If

Case 4 'maintainerAct
If maintainerAct = True Then
    iPageNumber = 7
    DoCmd.GoToControl "Page7"
    Me.cmdNext.Enabled = False
    Me.cmdBack.Enabled = True
Else
    Me.chkP18MaintActs.Value = True
    iFirstLevelCounter = 5
    With Me.lblP18MaintActs
        .ForeColor = QBColor(8)
        .FontWeight = 400
        .Caption = "COMPLETED - Input WORKING
CONDITIONS related factors."
    End With
    iPageNumber = 18
    DoCmd.GoToControl "Page18"
    'GoTo startSelect
End If

Case 5 'Done
'MsgBox "All factors should now be added. Click
next to continue.", vbOKOnly, "All Factors Added"
iPageNumber = 19
DoCmd.GoToControl "Page19"
Me.cmdNext.Enabled = False
Me.cmdFinish.Enabled = True

End Select

End Sub

```

'Function/Sub Name:

```

' - managementCond()
' - maintainerCond()
' - workingCond()
' - maintainerAct()

```

Description: 4 Functions. For prompting users for type of 1st level

'factor to input.

'

'Input: None

'

'Output: None

'

'References: None

'

Private Function managementCond() As Boolean

Dim response As Variant

```

    response = MsgBox("Was there a Management Condition
that contributed to this mishap?" & Chr(13) & Chr(13) & _
    "Examples:" & Chr(13) & _
    " - An engine change is performed despite a high sea
state." & Chr(13) & _
    " - A manual omits a step calling for an o-ring to be
installed." & Chr(13) & _
    " - A commander does not ensure that personnel wear
required protective gear." & Chr(13) & _
    " - A technical publication does not specify torque
requirements." & Chr(13) & _
    " - A poor component layout prohibits direct viewing
during inspection." & Chr(13) & Chr(13) & _
    "Click yes to enter a factor. No to go to the next
category.", vbYesNo + vbQuestion + vbDefaultButton1,
"First Level Factors")

```

If response = vbYes Then

managementCond = True

Else

managementCond = False

End If

End Function

Private Function maintainerCond() As Boolean

Dim response As Variant

```

    response = MsgBox("Was there a Maintainer Condition
that contributed to this mishap?" & Chr(13) & Chr(13) & _
    "Examples:" & Chr(13) & _
    " - A maintainer with life stress has impaired
concentration." & Chr(13) & _
    " - A maintainer is fatigued from working 20 hours
straight." & Chr(13) & _
    " - A short maintainer cannot visually inspect an
aircraft component." & Chr(13) & _
    " - A maintainer using improper hand signals." &
Chr(13) & _
    " - A maintainer signs off an inspections due to
perceived pressure." & Chr(13) & _
    " - A maintainer working on an aircraft skipped a
requisite training evolution." & Chr(13) & Chr(13) & _
    "Click yes to enter a factor. No to go to the next
category.", vbYesNo + vbQuestion + vbDefaultButton1,
"First Level Factors")

```

If response = vbYes Then

maintainerCond = True

Else

maintainerCond = False

End If

```

End Function
Private Function workingCond() As Boolean

    Dim response As Variant

    response = MsgBox("Was there a Working Condition that
contributed to this mishap?" & Chr(13) & Chr(13) & _
    "Examples:" & Chr(13) & _
    "  - A maintainer working at night without artificial
lighting." & Chr(13) & _
    "  - A maintainer securing an aircraft in a driving rain
improperly chocks a wheel" & Chr(13) & _
    "    working at night without artificial lighting." &
Chr(13) & _
    "  - A maintainer slips on a pitching deck." & Chr(13)
& _
    "  - A maintainer uses faulty test set." & Chr(13) & _
    "  - A maintainer in a fuel cell cannot reach a
component." & Chr(13) & _
    "  - A maintainer's view in spotting an aircraft is
obscured by catapult steam." & Chr(13) & Chr(13) & _
    "Click yes to enter a factor. No to go to the next
category.", vbYesNo + vbQuestion + vbDefaultButton1,
"First Level Factors")

    If response = vbYes Then
        workingCond = True
    Else
        workingCond = False
    End If

```

```

End Function
Private Function maintainerAct() As Boolean

```

```

    Dim response As Variant

    response = MsgBox("Was there a Maintainer Act that
contributed to this mishap?" & Chr(13) & Chr(13) & _
    "Examples:" & Chr(13) & _
    "  - A maintainer misses a hand signal." & Chr(13) & _
    "  - A maintainer inflates a tire using a pressure required
by a different aircraft." & Chr(13) & _
    "  - A maintainer misjudges the distance between a tow
tractor an aircraft wing." & Chr(13) & _
    "  - A maintainer engages in practices, condoned by
management, that bend the rules." & Chr(13) & _
    "  - A maintainer willfully breaks standing rules
disregarding the consequences." & Chr(13) & Chr(13) & _
    "Click yes to enter a factor. No to Finish.", vbYesNo +
vbQuestion + vbDefaultButton1, "First Level Factors")

    If response = vbYes Then
        maintainerAct = True
    Else
        maintainerAct = False
    End If

```

```

End Function

```

```

'=====
Function/Sub Name: addFactor()
'

```

```

'Description: Creates a new default factor.
'

```

```

'Input:
'  - s3rdLevelFactor    - Type of factor to create.
'  - sShortDescription  - Short description for the factor.
'

```

```

'
'Output: None
'
'References: None
'

```

```

Private Function addFactor(s3rdLevelFactor As String,
sShortDescription As String) As Boolean

```

```

    iFactorsAddedCounter = iFactorsAddedCounter + 1
    Me.txtFactorCounter.Value = iFactorsAddedCounter

```

```

'On Error GoTo StartError

```

```

    DoCmd.SetWarnings (False)
    DoCmd.RunSQL "INSERT INTO tblMishapFactors
(MishapID_FK, FactorSummary, 3rdLevelCode_FK)
VALUES ('" & GlobalDeclarations.gLngMishapToGet & "',
'" & sShortDescription & "', '" & s3rdLevelFactor & "');"
    DoCmd.SetWarnings (True)

```

```

    addFactor = True

```

```

ExitSub:

```

```

Exit Function

```

```

StartError:
    addFactor = False
    GoTo ExitSub

```

```

End Function

```

```

'=====
Function/Sub Name: cmdCancel_Click()
'

```

```

'Description: Closes the form undoing changes.
'

```

```

'Input: None
'

```

```

'Output: None
'

```

```

'References: None
'

```

```

Private Sub cmdCancel_Click()

```

```

    On Error GoTo Err_cmdCancel_Click

```

```

    GlobalDeclarations.gFormNeedsRefresh = True
    bBackToMain = True 'Have to use a flag to differentiate a
cancel from a finish
    DoCmd.Close acForm, "1-0-0-5-frm-addMishap"

```

```

Exit_cmdCancel_Click:
    Exit Sub

```

```

Err_cmdCancel_Click:
    MsgBox ERR.Description
    Resume Exit_cmdCancel_Click

```

```

End Sub

```

```

'=====
Function/Sub Name: addMishap()
'

```

```

'Description: Creates a new default Mishap.
'

```



```

'
'Input: None.
'
'Output: None
'
'References: None
'
'=====
Private Function addMishap() As Boolean

    On Error GoTo StartError

    DoCmd.SetWarnings (False)
    DoCmd.RunSQL "INSERT INTO tblMishaps
(MishapDate, Aircraft_FK, Class_FK, Type_FK,
LocationID_FK," & _
    "OrgID_FK, ShortDescription, LongDescription,
DatabaseType) VALUES (" & _
    Me.txtDate.Value & ", " & _
    Me.cboAircraftType.Value & ", " & _
    Me.cboClass.Value & ", " & _
    Me.cboType.Value & ", " & _
    Me.cboLocation.Value & ", " & _
    Me.cboOrganization.Value & ", " & _
    Me.txtShortDescription.Value & ", " & _
    Me.txtLongDescription.Value & ", " & _
    Me.txtDatabaseType.Value & ");"
    DoCmd.SetWarnings (True)

    'Now determine the MishapID that was just created by
    getting the max value
    Dim conn As New ADODB.Connection
    Dim rst As New ADODB.Recordset
    Dim sTempHolder As String

    'Open a connection to the data
    Set conn = Application.CurrentProject.Connection

    'Open a recordset with a keyset cursor
    rst.Open "SELECT max(MishapID) FROM tblMishaps",
conn, adOpenDynamic, adLockOptimistic, adCmdText

    rst.MoveFirst
    MsgBox rst.Fields(0)
    Me.TxtGlobalFocus.Value = rst.Fields(0)

    'Clean up
    rst.Close

```

```

conn.Close

addMishap = True

ExitSub:

Exit Function

StartError:
addMishap = False
GoTo ExitSub

End Function

'=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
'References:
' - clFormWindow
'
'=====
Public Sub MoveToCenter(ByVal strFormName As String)

    Dim fwForm As New clFormWindow

    With fwForm
        .hwnd = Forms(strFormName).hwnd
        .Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) *
0.6)
        .Left = (.Parent.Width - .Width) / 2
    End With
    Set fwForm = Nothing

End Sub

```

## **FORMCLASS-1-0-0-7-PopUpFrm-CodeMaintenance**

```
Option Compare Database
Option Explicit
#####
'          FORM DESCRIPTION
#####
'Class Name: 1-0-0-7-PopUpFrm-CodeMaintenance
'
'Author: Pat Flanders & Scott Tufts
'
'Allows an Administrator to add codes directly to the database
code
'lookup tables.
'
'References:
'    - tblAircraft
'    - tblMishapClass
'    - tblMishapLocation
'    - tblOrganization
'    - tblmishaptype
'#####

*****
'          FUNCTIONS
*****

=====
'Function/Sub Name: cmdClose_Click()
'
'Description: Closes the form.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub cmdClose_Click()
    DoCmd.Close acForm, "1-0-0-7-PopUpFrm-CodeMaintenance"
End Sub

=====
'Function/Sub Name: cmdOK_Click()
'
'Description: Opens the appropriate table for direct editing
based
'on the radio button selection in the frame.
'
'Input: None
'
'Output: None
'
'References: None
'
=====
Private Sub cmdOk_Click()

    If Me.Frame6 = 1 Then

        DoCmd.OpenForm "1-0-0-7-PopUpFrm-CodeMaint-
tblAircraft", acFormDS
        End If

        If Me.Frame6 = 2 Then
            DoCmd.OpenForm "1-0-0-7-PopUpFrm-CodeMaint-
tblMishapClass", acFormDS
            End If

        If Me.Frame6 = 3 Then
            DoCmd.OpenForm "1-0-0-7-PopUpFrm-CodeMaint-
tblMishapLocation", acNormal
            End If

        If Me.Frame6 = 4 Then
            DoCmd.OpenForm "1-0-0-7-PopUpFrm-CodeMaint-
tblOrganization", acNormal
            End If

        If Me.Frame6 = 5 Then
            DoCmd.OpenForm "1-0-0-7-PopUpFrm-CodeMaint-
tblMishapType", acFormDS
            End If

End Sub

=====
'Function/Sub Name: Form_Load()
'
'Description: Dynamically resizes the form to the users
screen
'resolution and then centers it.
'
'Input: None
'
'Output: None
'
'References:
'    - ezSizeForm
'
=====
Private Sub Form_Load()
    ezSizeForm Me, -1
    MoveToCenter "1-0-0-7-PopUpFrm-CodeMaintenance"
End Sub

=====
'Function/Sub Name: MoveToCenter()
'
'Description: Centers the form on the screen. Using the
ezSizeForm
'class breaks Access's built-in autocenter function, so this
'method is needed to fix it. Each form gets its own version of
this
'function so that minor adjustments can be made on a form by
form
'basis.
'
'Input: None
'
'Output: None
'
```

References:

' - clFormWindow

,

=====

Public Sub MoveToCenter(ByVal strFormName As String)

Dim fwForm As New clFormWindow

With fwForm

.hwnd = Forms(strFormName).hwnd

.Top = ((.Parent.Top - .Top) / 2) + ((.Parent.Top - .Top) \* 0.6)

.Left = (.Parent.Width - .Width) / 2

End With

Set fwForm = Nothing

End Sub

## **MODULE-DetermineOSDeclares**

Option Explicit

Type OSVERSIONINFO

dwOSVersionInfoSize As Long

dwMajorVersion As Long

dwMinorVersion As Long

dwBuildNumber As Long

dwPlatformId As Long

szCSDVersion As String \* 128 'Maintenance string for PSS usage

End Type

Declare Function GetVersionEx Lib "kernel32" Alias

"GetVersionExA" (lpVersionInformation As

OSVERSIONINFO) As Long

Declare Function GetSystemMetrics Lib "user32" (ByVal nIndex As Long) As Long

Public Const SM\_CLEANBOOT = 67

Public Const SM\_DEBUG = 22

Public Const SM\_SLOWMACHINE = 73

Public Const VER\_PLATFORM\_WIN32s = 0

Public Const VER\_PLATFORM\_WIN32\_WINDOWS = 1

Public Const VER\_PLATFORM\_WIN32\_NT = 2

#####  
' MODULE DESCRIPTION

#####  
'Class Name: DetermineOSDeclares.bas

'Author: Pat Flanders & Scott Tufts

'Description: Contains various functions for determining system properties like O/S type and version of Access that is running.

'The O/S type functions are declared above and result in direct querying of the Windows API.

'References: None

#####

\*\*\*\*\*  
' FUNCTIONS  
\*\*\*\*\*

=====

Function/Sub Name: IsRuntime()

'Description: Determines if Access runtime is being used to run the application. Access runtime has no support for reports.

'Input: None

'Output: Success or failure.

'References: None

=====

Function IsRuntime() As Boolean

' Check if this application is using the run-time version of Access.

IsRuntime = SysCmd(acSysCmdRuntime)

End Function

=====

Function/Sub Name: IsRunning()

'Description: To prevent a second instance from loading if a user mistakenly

'attempts to launch it twice. This code is called from the autoexec

'macro to test whether the app is already running and terminate

'the launch if a copy of it is already open.

'Input: None

'Output: -1 means that an instance is already running.

'References: None

=====

Function IsRunning() As Integer

If TestDDELink(Application.CurrentProject.Name) Then

'A -1 means that this is a second instance.

IsRunning = -1

Else

IsRunning = 0

End If

End Function

'Helper Function for IsRunning() above

Function TestDDELink(ByVal strAppName\$) As Integer

Dim varDDEChannel As Variant

On Error Resume Next

Application.SetOption ("Ignore DDE Requests"), True

varDDEChannel = DDEInitiate("MSAccess", strAppName)

'When the app isn't already running this will error

If ERR Then

TestDDELink = False

Else

TestDDELink = True

DDETerminate varDDEChannel

DDETerminateAll

End If

Application.SetOption ("Ignore DDE Requests"), False

End Function

## **MODULE-ezSizingFunctions**

```

Option Compare Database
Option Explicit
'#####
'      MODULE DESCRIPTION
'#####
'Class Name:  ezSizingFunctions.bas
'
'Author:  EZ Sizing Functions
'      Copyright (C) 2000 Database Creations, Inc.
'      Revision 6/14/00
'      based on 8/25/99 code with revisions
'
'Description:  Contains various functions for dynamically
resizing
'the forms in the application based on the user's screen
resolution.
'
'
'References:  None
'
'#####

*****
'      FUNCTIONS
*****
'Functions are defined below by the author and are Copyright
of
'Database Creations, Inc.

Type RECT
    x1 As Long
    y1 As Long
    x2 As Long
    y2 As Long
End Type

Type TEXTMETRIC
    tmHeight As Integer
    tmAscent As Integer
    tmDescent As Integer
    tmInternalLeading As Integer
    tmExternalLeading As Integer
    tmAveCharWidth As Integer
    tmMaxCharWidth As Integer
    tmWeight As Integer
    tmItalic As String * 1
    tmUnderlined As String * 1
    tmStruckOut As String * 1
    tmFirstChar As String * 1
    tmLastChar As String * 1
    tmDefaultChar As String * 1
    tmBreakChar As String * 1
    tmPitchAndFamily As String * 1
    tmCharSet As String * 1
    tmOverhang As Integer
    tmDigitizedAspectX As Integer
    tmDigitizedAspectY As Integer
End Type

Declare Function IsZoomed Lib "user32" (ByVal hwnd As
Long) As Long
Declare Function IsIconic Lib "user32" (ByVal hwnd As
Long) As Long

Declare Function GetDesktopWindow Lib "user32" () As
Long
Declare Function GetWindowRect Lib "user32" (ByVal
hwnd As Long, rectangle As RECT) As Long
Declare Function GetTextMetrics Lib "gdi32" Alias
"GetTextMetricsA" (ByVal hdc As Long, lpMetrics As
TEXTMETRIC) As Long
Declare Function GetWindowDC Lib "user32" (ByVal hwnd
As Long) As Long
Declare Function ReleaseDC Lib "user32" (ByVal hwnd As
Long, ByVal hdc As Long) As Long
Declare Function SetMapMode Lib "gdi32" (ByVal hdc As
Long, ByVal nMapMode As Long) As Long

Public Sub ezSizeForm(xForm As Form, ScaleFactor As
Single, Optional EchoOff As Boolean = True)
'This subroutine will resize the form specified by parameter
xForm by the factor of ScaleFactor
'If scale factor is 0 or negative, automatic scaling will occur
based on the following
' Value  Forms originally designed for
' 0      640 x 480
' -1     800 x 600
' -2     1024 x 768
' -3     1280 x 1024
' -4     1600 x 1200
' -5     1152 x 864 OR 1152 x 870

Dim ActiveForm As Object
Dim i As Integer
Dim D(200, 3) As Single

    On Error GoTo errorHandler
    If ScaleFactor = 1 Then GoTo Done
    If ScaleFactor <= 0 Then ScaleFactor =
ezGetScaleFactor(ScaleFactor)

    If EchoOff Then DoCmd.Echo False
    Set ActiveForm = xForm

    'If form in datasheet view then don't resize
    If xForm.CurrentView <> 1 Then GoTo Done

    'If the form is maximized then don't resize
    If IsZoomed(xForm.hwnd) <> 0 Then GoTo Done

    With ActiveForm
        If ScaleFactor > 1 Then 'form is growing
            'deal with section heights and form width first
            On Error Resume Next 'handle error for non-existent
sections
                For i = 0 To 4
                    .Section(i).Height = .Section(i).Height *
ScaleFactor
                Next i
            On Error GoTo errorHandler
            .Width = .Width * ScaleFactor
        End If

        'save old dimensions of subforms/groups/tabs
        For i = 0 To .Count - 1
            Select Case .Controls(i).ControlType
                Case acOptionGroup, acSubform, acTabCtl
                    D(i, 0) = .Controls(i).Width
                    D(i, 1) = .Controls(i).Height
            End Select
        Next i
    End With
End Sub

```

```

        D(i, 2) = .Controls(i).Left
        D(i, 3) = .Controls(i).Top
    End Select
Next i

'deal with controls
For i = 0 To .Count - 1
    Select Case .Controls(i).ControlType
        Case acOptionGroup, acPage
            'do nothing now
        Case acTabCtl
            .Controls(i).TabFixedWidth =
            .Controls(i).TabFixedWidth * ScaleFactor
            .Controls(i).TabFixedHeight =
            .Controls(i).TabFixedHeight * ScaleFactor
            If .Controls(i).Left < 0 Then .Controls(i).Left = 0
            .Controls(i).Left = .Controls(i).Left * ScaleFactor
            .Controls(i).Top = .Controls(i).Top * ScaleFactor
            .Controls(i).Width = .Controls(i).Width *
ScaleFactor
            .Controls(i).Height = .Controls(i).Height *
ScaleFactor
            .Controls(i).fontsize = .Controls(i).fontsize *
ScaleFactor
            Case acSubform
                On Error Resume Next
                ezSizeForm .Controls(i).Form, ScaleFactor,
False
                On Error GoTo errorHandler
            Case Else
                On Error Resume Next
                If .Controls(i).Left < 0 Then .Controls(i).Left = 0
                .Controls(i).Left = .Controls(i).Left *
ScaleFactor
                .Controls(i).Top = .Controls(i).Top *
ScaleFactor
                .Controls(i).Width = .Controls(i).Width *
ScaleFactor
                .Controls(i).Height = .Controls(i).Height *
ScaleFactor
                .Controls(i).fontsize = .Controls(i).fontsize *
ScaleFactor
                On Error GoTo errorHandler
            End Select
        Next i

'fix dimensions of subforms/groups/tabs
If ScaleFactor > 1 Then
    On Error Resume Next
    For i = 0 To 4
        .Section(i).Height = .Section(i).Height * ScaleFactor
    Next i
    On Error GoTo errorHandler
End If
For i = 0 To .Count - 1
    Select Case .Controls(i).ControlType
        Case acSubform
            .Controls(i).Width = D(i, 0) * ScaleFactor
            .Controls(i).Height = D(i, 1) * ScaleFactor
            .Controls(i).Left = D(i, 2) * ScaleFactor
            .Controls(i).Top = D(i, 3) * ScaleFactor
        End Select
    Next i
    For i = 0 To .Count - 1
        Select Case .Controls(i).ControlType
            Case acOptionGroup, acTabCtl
                .Controls(i).Left = D(i, 2) * ScaleFactor
                .Controls(i).Top = D(i, 3) * ScaleFactor

```

```

        .Controls(i).Width = D(i, 0) * ScaleFactor
        .Controls(i).Height = D(i, 1) * ScaleFactor
    End Select
Next i

'Resize form dimensions and fit window to form
On Error Resume Next
For i = 0 To 4
    .Section(i).Height = 0
Next i
On Error GoTo errorHandler
.Width = 0
DoCmd.RunCommand acCmdSizeToFitForm
GoTo Done

errorHandler:
    If ERR.Number = 2046 Then GoTo Done
    MsgBox "Error with control " & .Controls(i).Name &
vbCrLf & _
        "L: " & .Controls(i).Left & " - " & D(i, 2) & vbCrLf &
_
        "T: " & .Controls(i).Top & " - " & D(i, 3) & vbCrLf &
_
        "W: " & .Controls(i).Width & " - " & D(i, 0) &
vbCrLf & _
        "H: " & .Controls(i).Height & " - " & D(i, 1) &
vbCrLf
Done:
    If EchoOff Then DoCmd.Echo True
    End With

End Sub

Function ezGetScreenRes() As String
'This function returns the windows screen size
Dim R As RECT
Dim hwnd As Long
Dim RetVal As Long

    hwnd = GetDesktopWindow()
    RetVal = GetWindowRect(hwnd, R)
    ezGetScreenRes = (R.x2 - R.x1) & "x" & (R.y2 - R.y1)

End Function

Public Function ezGetScaleFactor(S) As Single
'Returns a scale factor for resizing based on the passed
parameter S
'which should represent the screen size a form was designed
for
'the scale factor returned is based on the current screen
resolution
    Select Case S
        Case 0 '640 x 480
            Select Case ezGetScreenRes
                Case "640x480"
                    ezGetScaleFactor = 1
                Case "800x600"
                    ezGetScaleFactor = 1.2
                Case "1024x768"
                    ezGetScaleFactor = 1.5
                Case "1152x864", "1152x870"
                    ezGetScaleFactor = 1.7
                Case "1280x1024"
                    ezGetScaleFactor = 1.9
                Case "1600x1200"
                    ezGetScaleFactor = 2.4

```

```

End Select
Case -1 '800 x 600
Select Case ezGetScreenRes
Case "640x480"
ezGetScaleFactor = 0.8
Case "800x600"
ezGetScaleFactor = 1
Case "1024x768"
ezGetScaleFactor = 1.2
Case "1152x864", "1152x870"
ezGetScaleFactor = 1.4
Case "1280x1024"
ezGetScaleFactor = 1.5
Case "1600x1200"
ezGetScaleFactor = 1.9
End Select
Case -2 '1024 x 768
Select Case ezGetScreenRes
Case "640x480"
ezGetScaleFactor = 0.6
Case "800x600"
ezGetScaleFactor = 0.7
Case "1024x768"
ezGetScaleFactor = 1
Case "1152x864", "1152x870"
ezGetScaleFactor = 1.05
Case "1280x1024"
ezGetScaleFactor = 1.1
Case "1600x1200"
ezGetScaleFactor = 1.4
End Select
Case -3 '1280 x 1024
Select Case ezGetScreenRes
Case "640x480"
ezGetScaleFactor = 0.5
Case "800x600"
ezGetScaleFactor = 0.6
Case "1024x768"
ezGetScaleFactor = 0.8
Case "1152x864", "1152x870"
ezGetScaleFactor = 0.9
Case "1280x1024"
ezGetScaleFactor = 1
Case "1600x1200"
ezGetScaleFactor = 1.1
End Select
Case -4 '1600 x 1200
Select Case ezGetScreenRes
Case "640x480"
ezGetScaleFactor = 0.3
Case "800x600"
ezGetScaleFactor = 0.4
Case "1024x768"
ezGetScaleFactor = 0.6
Case "1152x864", "1152x870"
ezGetScaleFactor = 0.65
Case "1280x1024"
ezGetScaleFactor = 0.7
Case "1600x1200"
ezGetScaleFactor = 1
End Select
Case -5 '1152 x 864 OR 1152 x 870
Select Case ezGetScreenRes
Case "640x480"
ezGetScaleFactor = 0.4
Case "800x600"
ezGetScaleFactor = 0.6
Case "1024x768"
ezGetScaleFactor = 0.8
Case "1152x864", "1152x870"
ezGetScaleFactor = 1
End Select
End Function
Public Function ezReSize(xForm As Form)
'This subroutine will resize the form based on it's current
dimensions
Dim ActiveForm As Object
Dim strTag As String
Dim SH As Single
Dim SW As Single

On Error GoTo errorHandler
Set ActiveForm = xForm

'If form in datasheet view then don't resize
If xForm.CurrentView <> 1 Then GoTo Done

'If the form is maximized then don't resize
If IsZoomed(xForm.hwnd) <> 0 Then GoTo Done

'If the form is minimized then don't resize
If IsIconic(xForm.hwnd) <> 0 Then GoTo Done

With ActiveForm
If .tag = "Sizing" Then GoTo Done
strTag = .tag
.tag = "Sizing"
'Determine size of window and set resize based on
lowest proportion
SH = .WindowHeight / .Section(0).Height
SW = .WindowWidth / .Width
If SH > SW Then
ezSizeForm xForm, SW
Else
ezSizeForm xForm, SH
End If
.Width = 0
On Error Resume Next
.tag = strTag
End With
GoTo Done
errorHandler:
MsgBox ERR.Description
Done:

End Function

Public Function ezLargeFonts() As Boolean
'This function returns a true if large fonts are being used.
Dim hdc As Long
Dim hwnd As Long
Dim PrevMapMode As Long
Dim tm As TEXTMETRIC

'Get the handle of the desktop window
hwnd = GetDesktopWindow()
'Get the device context for the desktop
hdc = GetWindowDC(hwnd)
If hdc Then 'Set the mapping mode to pixels

```

```
PrevMapMode = SetMapMode(hdc, 1)
'Get the size of the system font
GetTextMetrics hdc, tm
'Set the mapping mode back to what it was
PrevMapMode = SetMapMode(hdc, PrevMapMode)
'Release the device context
ReleaseDC hwnd, hdc
```

```
'If the system font is more than 16 pixels high, then
large fonts are being used
If tm.tmHeight > 16 Then ezLargeFonts = True Else
ezLargeFonts = False
End If

End Function
```



## **MODULE-GlobalDeclarations**

```

Option Compare Database
Option Explicit
'#####
'      MODULE DESCRIPTION
'#####
'Class Name: GlobalDeclarations.bas
'
'Author: Pat Flanders & Scott Tufts
'
'Description: Contains all definitions for application global
'variables. Most of these are needed due to the inability of
'VBA to pass parameters as part of a constructor.
'
'References: None
'
'#####

Global gLngMishapToGet As Long
Global gFormNeedsRefresh As Boolean
Global gBlnAddAMishap As Boolean
Global gStrDescription As String
Global gstrDatabaseType As String

'#####
'*****
'      FUNCTIONS
'*****
'#####

'Function/Sub Name: getDBType()
'
'Description: Determines the type of database (military or
'civilian)
'based on the SQL serverer tblDatabaseType settings.
'
'Input: None
'
'Output: None
'
'References: None
'
'#####
Public Sub getDBType()

    Dim conn As New ADODB.Connection
    Dim rst As New ADODB.Recordset
    Dim sTempHolder As String

    'Open a connection to the data
    Set conn = Application.CurrentProject.Connection

    'Open a recordset with a keyset cursor
    rst.Open "SELECT * FROM tblDatabaseType", conn,
    adOpenDynamic, adLockOptimistic, adCmdText

    'Walk the recordset
    Do Until rst.EOF
        If rst.Fields(0) = "M" Then sTempHolder = "M"
        rst.MoveNext
    Loop

    If sTempHolder = "M" Then

        GlobalDeclarations.gstrDatabaseType = "M"
    Else
        GlobalDeclarations.gstrDatabaseType = "C"
    End If

    'Clean up
    rst.Close
    conn.Close

End Sub

'=====
'Function/Sub Name: toggleDBType()
'
'Description: Toggles the current investigation module DB
'type.
'
'Input: None
'
'Output: Success or failure.
'
'References: None
'
'=====
Public Function toggleDBType() As Boolean

    On Error GoTo StartError
    GlobalDeclarations.getDBType

    DoCmd.SetWarnings (False)

    If GlobalDeclarations.gstrDatabaseType = "M" Then
        DoCmd.RunSQL "UPDATE tblDatabaseType SET
        tblDatabaseType.DatabaseType = " & Chr(34) & "C" &
        Chr(34) & " WHERE tblDatabaseType.DatabaseType=" &
        Chr(34) & "M" & Chr(34) & ";"
        GlobalDeclarations.gstrDatabaseType = "C"
    Else
        DoCmd.RunSQL "UPDATE tblDatabaseType SET
        tblDatabaseType.DatabaseType = " & Chr(34) & "M" &
        Chr(34) & " WHERE tblDatabaseType.DatabaseType=" &
        Chr(34) & "C" & Chr(34) & ";"
        GlobalDeclarations.gstrDatabaseType = "M"
    End If
    DoCmd.SetWarnings (True)

    Forms![1-0-0-0-frm-SelectMishap].Refresh

    toggleDBType = True

ExitSub:

    Exit Function

StartError:
    toggleDBType = False
    GoTo ExitSub

End Function

'=====
'Function/Sub Name: getDBTypeFromFile()
'

```

'Description: Determinese the type of database (military or civilian)  
'based on the HFACS.ini file settings.

'Input: None

'Output: Success or failure.

'References: None

---

Public Function getDbTypeFromFile() As Boolean

Dim sFileName As String  
Dim oINIFile As INIFile  
Set oINIFile = New INIFile  
oINIFile.Init ("HFACS")

On Error GoTo StartError  
Screen.MousePointer = 11  
Debug.Print "Reading ini data . . ."

' Get name for .ini file in the SYSTEM directory  
sFileName = oINIFile.GetIniFileName

Debug.Print sFileName

' Read values from .ini file. Specify file name, section, and key.

GlobalDeclarations.gstrDatabaseType =  
oINIFile.ReadFromIniFile(sFileName, \_  
"DBTYPE", "DBtype")

Debug.Print "Just read in " &  
GlobalDeclarations.gstrDatabaseType

Screen.MousePointer = 0  
If Trim(GlobalDeclarations.gstrDatabaseType) = "" Then  
getDbTypeFromFile = False  
Else  
getDbTypeFromFile = True  
End If

ExitSub:  
Set oINIFile = Nothing  
Exit Function

StartError:  
Screen.MousePointer = 0  
getDbTypeFromFile = False  
Resume ExitSub

End Function

---

'Function/Sub Name: synchFileDbTypeToDbValue()

'Description: Ensures that this program opens in the same mode (civilian  
'or military) as the HFACS instance that launched it.

'Input: None

'Output: None

'References: None

---

Public Sub synchFileDbTypeToDbValue()

Dim sTempNameHolder As String  
If GlobalDeclarations.getDbTypeFromFile = True Then  
sTempNameHolder =  
GlobalDeclarations.gstrDatabaseType  
GlobalDeclarations.getDbType  
If Trim(sTempNameHolder) <>  
GlobalDeclarations.gstrDatabaseType Then  
GlobalDeclarations.toggleDbType  
Else  
MsgBox "No ini file to read."  
End If

ExitSub:  
Exit Sub

StartError:  
GoTo ExitSub

End Sub

## APPENDIX L. MODIFIED VB SETUP1

### CLASS-INIFile

Option Explicit

```
#####  
' CLASS DESCRIPTION  
#####  
'Class Name: INIFile.cls  
'  
'Author: Microsoft Corporation. Modified by Pat Flanders  
&  
' Scott Tufts  
'  
'This class creates .ini File objects used to create, delete, set,  
'and get values in a standard format Microsoft .ini file. It  
uses  
'calls to the Windows API for efficiency.  
'  
'References: Windows API  
'  
'NOTE: See function headers for internal component  
references.  
#####
```

```
*****  
' PROPERTIES  
*****
```

```
'The name of the ini file to read  
##ModelId=3B294CFD03A9  
Private msWbkName As String
```

```
'API Wrapper Code - provided by Microsoft  
##ModelId=3B294CFE0000  
Private Declare Function WritePrivateProfileString Lib  
"kernel32" Alias "WritePrivateProfileStringA" (ByVal  
lpApplicationName As String, ByVal lpKeyName As String,  
ByVal lpString As String, ByVal lpFileName As String) As  
Long
```

```
##ModelId=3B294CFE00AB  
Private Declare Function GetPrivateProfileString Lib  
"kernel32" Alias "GetPrivateProfileStringA" (ByVal  
lpApplicationName As String, ByVal lpKeyName As Any,  
ByVal lpDefault As String, ByVal lpReturnedString As  
String, ByVal nSize As Long, ByVal lpFileName As String)  
As Long
```

```
##ModelId=3B294CFE0196  
Private Declare Function GetWindowsDirectory Lib  
"kernel32" Alias "GetWindowsDirectoryA" (ByVal lpBuffer  
As String, ByVal nSize As Long) As Long
```

```
*****  
' FUNCTIONS  
*****
```

```
=====
```

Function/Sub Name: Init()

Description: If an instance of a class is created using the  
psuedo-  
'constructors from the Constructors.bas module, this function  
is  
'called to pass initial values, thereby mimicking the behavior  
of  
'a constructor with arguments. Passed in values are all  
required, but  
'the Constructors.New\_INIFile() function automatically sets  
'passed-in values to global variable values if they are left  
'blank.

Input:  
' sPassedInWorkBookName - Name of the .ini file to  
manipulate

Output: None

References:  
' - Constructors.bas

```
=====
```

```
##ModelId=3B294CFE0213  
Friend Sub Init(sPassedInWorkBookName As String)
```

```
msWbkName = sPassedInWorkBookName
```

```
End Sub
```

```
=====
```

Function/Sub Name: WriteToIniFile()

Description: Write a section, key, and value to an .ini file.

Input:  
' strSection - Name of a section  
' strKey - Name of a key  
' strValue - Name of a key value  
' strFileName - Name of the file to manipulate

Output: Success or failure

References: None

```
=====
```

```
##ModelId=3B294CFE0251  
Friend Function WriteToIniFile(strSection As String, strKey  
As String, strValue As String, strFileName As String) As  
Boolean
```

```
' Pass in name of section, key, key value, and file name.  
If WritePrivateProfileString(strSection, strKey, _  
strValue, strFileName) Then  
WriteToIniFile = True  
Else  
MsgBox "Error writing to .ini file: " & Err.LastDllError  
WriteToIniFile = False  
End If
```

End Function

```
=====
Function/Sub Name: DeleteIniSection()
'
'Description: Delete a section and all of its keys from an .ini
file.
'
'Input:
' strSection - Name of a section
' strFileName - Name of the file to manipulate
'
'Output: Success or failure
'
'References: None
=====
##ModelId=3B294CFE02DE
Friend Function DeleteIniSection(strSection As String,
strFileName As String) As Boolean

    If WritePrivateProfileString(strSection, vbNullString, _
        vbNullString, strFileName) Then
        DeleteIniSection = True
    Else
        MsgBox "Error deleting section from .ini file: " _
            & Err.LastDllError
        DeleteIniSection = False
    End If
End Function
```

End Function

```
=====
Function/Sub Name: DeleteIniKey()
'
'Description: Delete a key and its value from an .ini file.
'
'Input:
' strSection - Name of a section
' strKey - Name of a key
' strFileName - Name of the file to manipulate
'
'Output: Success or failure
'
'References: None
=====
##ModelId=3B294CFE033C
Friend Function DeleteIniKey(strSection As String, strKey
As String, strFileName As String) As Boolean

    If WritePrivateProfileString(strSection, strKey, _
        vbNullString, strFileName) Then
        DeleteIniKey = True
    Else
        MsgBox "Error deleting section from .ini file: " _
            & Err.LastDllError
        DeleteIniKey = False
    End If
End Function
```

End Function

```
=====
Function/Sub Name: GetIniFileName()
'
```

'Description: Return name for .ini file. Name includes name of  
'workbook file and ".ini". File path can be made the Windows  
directory.

'by uncommenting the code below

'Input: None

'Output: String path (e.g. C:\windows\HFACS.ini).

'References: None

```
=====
##ModelId=3B294CFE03A9
Friend Function GetIniFileName() As String
```

```
    Dim strWinDir As String
    Dim lngLen As Long
```

```
    ' Create null-terminated string to pass to
    ' GetWindowsDirectory.
    strWinDir = String$(255, vbNullChar)
```

```
    lngLen = Len(strWinDir)
```

```
    ' Return Windows directory.
    GetWindowsDirectory strWinDir, lngLen
```

```
    ' Truncate before first null character.
    strWinDir = Left(strWinDir, _
        InStr(strWinDir, vbNullChar) - 1)
```

```
    ' Return .ini file name.
    GetIniFileName = strWinDir & "\" & msWbkName &
".ini"
```

```
GetIniFileName = App.Path & "\" & msWbkName & ".ini"
```

End Function

```
=====
Function/Sub Name: ReadFromIniFile()
'
'Description: Read a value from an .ini file, given the file
name,
'section, key, and default value to return if key is not found.
'
'Input:
' strSection - Name of a section
' strKey - Name of a key
' strDefault - Default name of a key value
' strFileName - Name of the file to manipulate
'
'Output: Success or failure
'
'References: None
=====
##ModelId=3B294CFE03D8
Friend Function ReadFromIniFile(strFileName As String,
strSection As String, strKey As String, Optional strDefault
As String = "") As String
```

'Output: Success or failure

'References: None

```
=====
##ModelId=3B294CFE03D8
Friend Function ReadFromIniFile(strFileName As String,
strSection As String, strKey As String, Optional strDefault
As String = "") As String
```

```
    Dim strValue As String
```

```
    ' Fill string buffer with null characters.
    strValue = String$(255, vbNullChar)
```

```
    ' Attempt to read value. GetPrivateProfileString
```

```

' function returns number of characters written
' into string.
If GetPrivateProfileString(strSection, strKey, _
    strDefault, strValue, Len(strValue), _
    strFileName) > 0 Then
    ' If characters have been written into string, parse string
    ' and return.
    strValue = Left(strValue, InStr(strValue, vbNullChar) -
1)

    ReadFromIniFile = strValue
Else
    ' Otherwise, return a zero-length string.
    ReadFromIniFile = strDefault
End If

End Function

```

## **SETUP 1 Modification Code**

The Package & Deployment Wizard "Setup1.exe" program requires modification for use with the HFACS program. There are two areas that are modified:

- 1) Code for updating the HFACS.ini file with the location that the user installs HFACS. This is how HFACS knows where to look for its components.
- 2) Code for adding the HFACS Icon (rather than the MS Access Icon) to the START menu bar in Windows. This provides a more professional appearance.

The following section outlines the modifications needed for item 1 above. Item 2 instructions can be found in Microsoft Knowledgebase article Q240965.

### **MAKE THE FOLLOWING CHANGES TO THE STANDARD SETUP1.VBP:**

#### **Step1 - Modifications to basSetup1**

'Added by Pat Flanders for use with INIFile Class and copy.frm Unload event  
Global myAppPath As String

#### **Step2 – Modifications to frmBegin**

Private Sub Form\_QueryUnload(Cancel As Integer, UnloadMode As Integer)

```
****
' Set a global variable for the app.path
basSetup1.myAppPath = Me.lblDestDir.Caption
HandleFormQueryUnload UnloadMode, Cancel, Me
End Sub
```

#### **Step3 – Modifications to frmCopy**

Private Sub Form\_Unload(Cancel As Integer)

```
*****
' Now that the files have been copied, write an entry to the
iniFile for app.path
Dim theINIFile As INIFile
Dim strFileName As String
Dim writeSuccess As Boolean
Set theINIFile = New INIFile
strFileName = theINIFile.GetIniFileName
' Attempt to write values to .ini file. Specify
' file name, section, and key.
writeSuccess =
theINIFile.WriteToIniFile("CONNECTION",
"InstallDir", basSetup1.myAppPath, strFileName)
Set theINIFile = Nothing

End Sub
```

#### **Step4 – Import the INIFile class used in the HFACS Connection Component.**

## APPENDIX M. STORED PROCEDURES

### *1-0-0-1-flanAllMishapsByDate*

Alter Procedure [1 -0-0-1 -flanAllMishapsByDate]

```
(  
    @MishapID          int          = NULL  
)
```

As  
set nocount on

```
SELECT      MishapID,  
            MishapDate,  
            Aircraft_FK,  
            Class_FK,  
            tblMishapClass.MishapClassDefinition,  
            Type_FK,  
            tblMishapType.MishapTypeDefinition,  
            LocationID_FK,  
            tblMishapLocation.MishapLocation,  
            OrgID_FK,  
            tblOrganization.OrgName,  
            ShortDescription,  
            LongDescription ,  
            tblDatabaseType.DatabaseType  
  
FROM          (tblDatabaseType INNER JOIN tblMishaps ON tblDatabaseType.DatabaseType = tblMishaps.DatabaseType)  
LEFT JOIN tblMishapLocation ON tblMishaps.LocationID_FK = tblMishapLocation.MishapLocationID  
LEFT JOIN tblMishapClass ON tblMishaps.Class_FK = tblMishapClass.MishapClassCode  
LEFT JOIN tblMishapType ON tblMishaps.Type_FK = tblMishapType.MishapTypeCode  
LEFT JOIN tblOrganization ON tblMishaps.OrgID_FK = tblOrganization.OrgID  
  
WHERE      MishapID=COALESCE(@MishapID,          tblMishaps.MishapID)          and  
tblMishaps.DatabaseType=tblDatabaseType.DatabaseType  
ORDER BY      MishapDate  
  
return
```

### **1-0-0-2-flanAllMishapFactorsByID**

```
Alter Procedure [1-0-0-2-flanAllMishapFactorsByID]
(
    @MishapID_FK          int          = NULL
)
As
set nocount on

SELECT  tblMishapFactors.FactorID,
        tblMishapFactors.MishapID_FK,
        tblMishapFactors.FactorSummary,
        tblMishapFactors.[3rdLevelCode_FK],
        tblFactors.[3rdLevelDesc],
        tblFactors.[2ndLevelCode],
        tblFactors.[2ndLevelDesc],
        tblFactors.[1stLevelCode],
        tblFactors.[1stLevelDesc]

FROM      tblMishapFactors LEFT JOIN tblFactors ON tblMishapFactors.[3rdLevelCode_FK] =
tblFactors.[3rdLevelCode]

WHERE      MishapID_FK=@MishapID_FK

ORDER BY  tblMishapFactors.FactorID

return
```



### **1-0-0-3-flanInsertFactor**

```
Alter Procedure [1-0-0-3-flanInsertFactor]
(
    @MishapID int
)
As
set nocount on

Insert into tblMishapFactors (MishapID_FK, FactorSummary,[3rdLevelCode_FK])
Values (@MishapID, 'Please enter a summary','UNK')

return
```

### **1-0-0-4-flanIsUserSysadmin**

Alter Procedure [1-0-0-4-flanIsUserSysadmin]

As

DECLARE @IsAdmin int

SELECT IS\_SRVROLEMEMBER('sysadmin') as IsUserOwner

return

## **2-0-0-1-flanCountflanFilteredMishaps**

Alter Procedure [2-0-1-1-flanCountflanFilteredMishaps]

```
(
    @AC          varchar(10) = NULL, --default value is NULL for all parameters not specified
    @Type        varchar(3) = NULL,
    @Class       varchar(1) = NULL,
    @Loc         varchar(25) = NULL,
    @Svc         varchar(10) = NULL,
    @Yr          datetime = NULL,
    @1stLevel    varchar(5) = NULL,
    @2ndLevel    varchar(5) = NULL,
    @3rdLevel    varchar(5) = NULL
)

As

SELECT count(dbo.tblMishaps.MishapID) as NumRecords

FROM    dbo.tblDatabaseType INNER JOIN
        dbo.tblMishapLocation INNER JOIN
        dbo.tblMishapType INNER JOIN
        dbo.tblMishaps ON
        dbo.tblMishapType.MishapTypeCode = dbo.tblMishaps.Type_FK INNER JOIN
        dbo.tblMishapClass ON
        dbo.tblMishaps.Class_FK = dbo.tblMishapClass.MishapClassCode ON
        dbo.tblMishapLocation.MishapLocationID = dbo.tblMishaps.LocationID_FK INNER JOIN
        dbo.tblOrganization ON
        dbo.tblMishaps.OrgID_FK = dbo.tblOrganization.OrgID ON
        dbo.tblDatabaseType.DatabaseType = dbo.tblMishaps.DatabaseType

WHERE   dbo.tblMishaps.Aircraft_FK = COALESCE(@AC, dbo.tblMishaps.Aircraft_FK) AND
        dbo.tblMishaps.Type_FK = COALESCE(@Type, dbo.tblMishaps.Type_FK) AND
        dbo.tblMishaps.Class_FK = COALESCE(@Class, dbo.tblMishaps.Class_FK) AND
        dbo.tblMishaps.LocationID_FK = COALESCE(@Loc, dbo.tblMishaps.LocationID_FK) AND
        dbo.tblMishaps.OrgID_FK = COALESCE(@Svc, dbo.tblMishaps.OrgID_FK) AND
        datepart(year,dbo.tblMishaps.MishapDate) = COALESCE(@Yr, datepart(year,dbo.tblMishaps.MishapDate))

return
```

## **2-0-0-1-flanFilteredMishapTable**

Alter Procedure [2-0-1-1-flanFilteredMishapTable]

```
(
    @AC          varchar(10) = NULL, --default value is NULL for all parameters not specified
    @Type        varchar(3) = NULL,
    @Class       varchar(1) = NULL,
    @Loc         varchar(25) = NULL,
    @Svc         varchar(10) = NULL,
    @Yr          int = NULL,
    @1stLevel    varchar(5) = NULL,
    @2ndLevel    varchar(5) = NULL,
    @3rdLevel    varchar(5) = NULL
)

As
set nocount on

SELECT dbo.tblMishaps.MishapID, dbo.tblMishaps.MishapDate,
       dbo.tblMishaps.Aircraft_FK, dbo.tblMishaps.Class_FK,
       dbo.tblMishapClass.MishapClassDefinition, dbo.tblMishaps.Type_FK,
       dbo.tblMishapType.MishapTypeDefinition,
       dbo.tblMishaps.LocationID_FK, dbo.tblMishapLocation.MishapLocation,
       dbo.tblMishaps.OrgID_FK, dbo.tblOrganization.OrgName,
       dbo.tblMishaps.ShortDescription, dbo.tblMishaps.LongDescription,
       dbo.tblMishaps.DatabaseType
FROM   dbo.tblDatabaseType INNER JOIN
       dbo.tblMishapLocation INNER JOIN
       dbo.tblMishapType INNER JOIN
       dbo.tblMishaps ON
       dbo.tblMishapType.MishapTypeCode = dbo.tblMishaps.Type_FK INNER JOIN
       dbo.tblMishapClass ON
       dbo.tblMishaps.Class_FK = dbo.tblMishapClass.MishapClassCode ON
       dbo.tblMishapLocation.MishapLocationID = dbo.tblMishaps.LocationID_FK INNER JOIN
       dbo.tblOrganization ON
       dbo.tblMishaps.OrgID_FK = dbo.tblOrganization.OrgID ON
       dbo.tblDatabaseType.DatabaseType = dbo.tblMishaps.DatabaseType

WHERE  dbo.tblMishaps.Aircraft_FK = COALESCE(@AC, dbo.tblMishaps.Aircraft_FK) AND
       dbo.tblMishaps.Type_FK = COALESCE(@Type, dbo.tblMishaps.Type_FK) AND
       dbo.tblMishaps.Class_FK = COALESCE(@Class, dbo.tblMishaps.Class_FK) AND
       dbo.tblMishaps.LocationID_FK = COALESCE(@Loc, dbo.tblMishaps.LocationID_FK) AND
       dbo.tblMishaps.OrgID_FK = COALESCE(@Svc, dbo.tblMishaps.OrgID_FK) AND
       datepart(year, dbo.tblMishaps.MishapDate) = COALESCE(@Yr, datepart(year, dbo.tblMishaps.MishapDate))

return
```

## 2-0-2-1-flanSummaryGetNumbers

Alter Procedure [2-0-2-1 -flanSummaryGetNumbers]

```
(
    @AC_Type          varchar(10) = NULL,
    @Mishap_Type      varchar(3) = NULL,
    @Mishap_Class     varchar(1) = NULL,
    @Location         varchar(25) = NULL,
    @Service          varchar(10) = NULL,
    @Year             int          = NULL,
    @1stLevel         varchar(5) = NULL,
    @2ndLevel         varchar(5) = NULL,
    @3rdLevel         varchar(5) = NULL
)

As

Set nocount on

--Insert filtered data into Temp Filter_Table

SELECT DISTINCT dbo.tblMishaps.MishapID INTO
#Result

FROM    dbo.tblMishaps INNER JOIN
        dbo.tblDatabaseType ON
        dbo.tblMishaps.DatabaseType=dbo.tblDatabaseType.Database
eType INNER JOIN
        dbo.tblMishapFactors ON
        dbo.tblMishaps.MishapID =
        dbo.tblMishapFactors.MishapID_FK INNER JOIN
        dbo.tblFactors ON
        dbo.tblMishapFactors.[3rdLevelCode_FK] =
        dbo.tblFactors.[3rdLevelCode]

WHERE   dbo.tblMishaps.Aircraft_FK =
        COALESCE(@AC_Type, dbo.tblMishaps.Aircraft_FK)
AND
        dbo.tblMishaps.Type_FK =
        COALESCE(@Mishap_Type, dbo.tblMishaps.Type_FK)
AND
        dbo.tblMishaps.Class_FK =
        COALESCE(@Mishap_Class, dbo.tblMishaps.Class_FK)
AND
        dbo.tblMishaps.LocationID_FK =
        COALESCE(@Location, dbo.tblMishaps.LocationID_FK)
AND
        dbo.tblMishaps.OrgID_FK =
        COALESCE(@Service, dbo.tblMishaps.OrgID_FK) AND
        datepart(year, dbo.tblMishaps.MishapDate) =
        COALESCE(@Year,
        datepart(year, dbo.tblMishaps.MishapDate)) AND
        dbo.tblFactors.[1stLevelCode] =
        COALESCE(@1stLevel, dbo.tblFactors.[1stLevelCode])
AND
        dbo.tblFactors.[2ndLevelCode] =
        COALESCE(@2ndLevel, dbo.tblFactors.[2ndLevelCode])
AND
        dbo.tblFactors.[3rdLevelCode] =
        COALESCE(@3rdLevel, dbo.tblFactors.[3rdLevelCode])
AND
        dbo.tblMishaps.DatabaseType=dbo.tblDatabaseType.Database
eType

-----Build MishapCount resultset -----
SELECT
```

```
(
    SELECT Count([MishapID])
    FROM #Result
    WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND
        dbo.tblFactors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (
        dbo.tblFactors.[1stLevelCode] = 'MG')))) AS
    MG,
    (
    SELECT Count([MishapID])
    FROM #Result
    WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND
        dbo.tblFactors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (
        dbo.tblFactors.[1stLevelCode] = 'MC')))) AS
    MC,
    (
    SELECT Count([MishapID])
    FROM #Result
    WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND
        dbo.tblFactors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (
        dbo.tblFactors.[1stLevelCode] = 'WC')))) AS
    WC,
    (
    SELECT Count([MishapID])
    FROM #Result
    WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND
        dbo.tblFactors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (
        dbo.tblFactors.[1stLevelCode] = 'MA')))) AS
    MA,
    (
```

```

SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND

        dbo.tblFact ors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (

        dbo.tblFactors.[2ndLevelCode] = 'ORG'))))) AS
ORG,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND

        dbo.tblFactors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (

        dbo.tblFactors.[2ndLevelCode] = 'SUP'))))) AS
SUP,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND

        dbo.tblFactors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (

        dbo.tblFactors.[2ndLevelCode] = 'MED'))))) AS
MED,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND

        dbo.tblFactors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (

        dbo.tblFactors.[2ndLevelCode] = 'CRW'))))) AS
CRW,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (

```

```

        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND

        dbo.tblFactors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (

        dbo.tblFactors.[2ndLevelCode] = 'RDY'))))) AS
RDY,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND

        dbo.tblFactors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (

        dbo.tblFactors.[2ndLevelCode] = 'ENV'))))) AS
ENV,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND

        dbo.tblFactors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (

        dbo.tblFactors.[2ndLevelCode] = 'EQP'))))) AS
EQP,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors
        WHERE #Result.MishapID =
        dbo.tblMishapFactors.MishapID_FK AND

        dbo.tblFactors.[3rdLevelCode] =
        dbo.tblMishapFactors.[3rdLevelCode_FK] AND (

        dbo.tblFactors.[2ndLevelCode] = 'WRK'))))) AS
WRK,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
        #Result.MishapID
        FROM #Result, dbo.tblFactors,
        dbo.tblMishapFactors

```

```

WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND

    dbo.tblFactors.[3rdLevelCode] =
dbo.tblMishapFactors.[3rdLevelCode_FK] AND (

    dbo.tblFactors.[2ndLevelCode] = 'ERR')))) AS
ERR,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
SELECT DISTINCT
#Result.MishapID
FROM #Result, dbo.tblFactors,
dbo.tblMishapFactors
WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND

    dbo.tblFactors.[3rdLevelCode] =
dbo.tblMishapFactors.[3rdLevelCode_FK] AND (

    dbo.tblFactors.[2ndLevelCode] = 'VIO')))) AS
VIO,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
SELECT DISTINCT
#Result.MishapID
FROM #Result, dbo.tblMishapFactors
WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

    dbo.tblMishapFactors.[3rdLevelCode_FK] =
'PRO')))) AS PRO,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
SELECT DISTINCT
#Result.MishapID
FROM #Result, dbo.tblMishapFactors
WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

    dbo.tblMishapFactors.[3rdLevelCode_FK] =
'DOC')))) AS DOC,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
SELECT DISTINCT
#Result.MishapID
FROM #Result, dbo.tblMishapFactors
WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

    dbo.tblMishapFactors.[3rdLevelCode_FK] =
'DES')))) AS DES,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
SELECT DISTINCT
#Result.MishapID
FROM #Result, dbo.tblMishapFactors

```

```

WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

    dbo.tblMishapFactors.[3rdLevelCode_FK] =
'RES')))) AS RES,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
SELECT DISTINCT
#Result.MishapID
FROM #Result, dbo.tblMishapFactors
WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

    dbo.tblMishapFactors.[3rdLevelCode_FK] =
'IDQ')))) AS IDQ,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
SELECT DISTINCT
#Result.MishapID
FROM #Result, dbo.tblMishapFactors
WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

    dbo.tblMishapFactors.[3rdLevelCode_FK] =
'OPS')))) AS OPS,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
SELECT DISTINCT
#Result.MishapID
FROM #Result, dbo.tblMishapFactors
WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

    dbo.tblMishapFactors.[3rdLevelCode_FK] =
'PRB')))) AS PRB,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
SELECT DISTINCT
#Result.MishapID
FROM #Result, dbo.tblMishapFactors
WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

    dbo.tblMishapFactors.[3rdLevelCode_FK] =
'MIS')))) AS MIS,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
SELECT DISTINCT
#Result.MishapID
FROM #Result, dbo.tblMishapFactors
WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

    dbo.tblMishapFactors.[3rdLevelCode_FK] =
'MNT')))) AS MNT,
(
SELECT Count([MishapID])
FROM #Result

```







```

WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

        dbo.tblMishapFactors.[3rdLevelCode_FK] =
'IFC')))) AS IFC,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (
        SELECT DISTINCT
#Result.MishapID
        FROM #Result, dbo.tblMishapFactors
        WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

        dbo.tblMishapFactors.[3rdLevelCode_FK] =
'FLG')))) AS FLG,
(
SELECT Count([MishapID])
FROM #Result
WHERE (((#Result.MishapID) In (

```

```

SELECT DISTINCT
#Result.MishapID
FROM #Result, dbo.tblMishapFactors
WHERE #Result.MishapID =
dbo.tblMishapFactors.MishapID_FK AND (

        dbo.tblMishapFactors.[3rdLevelCode_FK] =
'EXC')))) AS EXC,
(
SELECT Count([#Result].[MishapID])
FROM #Result)
AS TotalMishaps;

return

```

## 2-0-2-1-flanSummaryGetRecords

```
Alter Procedure [2-0-2-1 -flanSummaryGetRecords]
(
    @AC          varchar(10) =
NULL,
    @Type        varchar(3) =
NULL,
    @Class       varchar(1) =
NULL,
    @Loc         varchar(25)=
NULL,
    @Svc         varchar(10)=
NULL,
    @Yr          int      =
NULL,
    @1stLevel    varchar(5) =
NULL,
    @2ndLevel    varchar(5) =
NULL,
    @3rdLevel    varchar(5) =
NULL
)
```

As  
set nocount on

```
SELECT DISTINCT dbo.tblMishapFactors.[MishapID_FK]
INTO #Result
```

```
FROM   dbo.tblMishapFactors INNER JOIN
dbo.tblFactors ON
dbo.tblMishapFactors.[3rdLevelCode_FK] =
dbo.tblFactors.[3rdLevelCode]
```

```
WHERE  dbo.tblFactors.[1stLevelCode] =
COALESCE(@1stLevel, dbo.tblFactors.[1stLevelCode])
AND
    dbo.tblFactors.[2ndLevelCode] =
COALESCE(@2ndLevel, dbo.tblFactors.[2ndLevelCode])
AND
    dbo.tblFactors.[3rdLevelCode] =
COALESCE(@3rdLevel, dbo.tblFactors.[3rdLevelCode])
```

```
ORDER BY dbo.tblMishapFactors.[MishapID_FK]
```

---Inner Query

```
SELECT
```

```
#Result.[MishapID_FK] ,
dbo.tblMishaps.[MishapID],
dbo.tblMishaps.[MishapDate],
dbo.tblMishaps.[Aircraft_FK],
dbo.tblMishaps.[Class_FK],
dbo.tblMishaps.[Type_FK],
```

```
dbo.tblMishaps.[LocationID_FK],
dbo.tblMishaps.[OrgID_FK],
dbo.tblMishaps.[DatabaseType],
dbo.tblMishaps.[ShortDescription],
dbo.tblMishaps.[LongDescription],
dbo.tblMishapClass .[MishapClassCode],
dbo.tblMishapClass .[MishapClassDefinition],
dbo.tblMishapLocation.[MishapLocationID],
dbo.tblMishapLocation.[MishapLocation],
dbo.tblMishapType .[MishapTypeCode],
dbo.tblMishapType .[MishapTypeDefinition],
dbo.tblOrganization .[OrgID],
dbo.tblOrganization .[OrgName]
```

```
FROM   #Result LEFT JOIN
        dbo.tblMishaps ON
dbo.tblMishaps.MishapID=#Result.[MishapID_FK] INNER
JOIN
        dbo.tblDatabaseType ON
dbo.tblMishaps.DatabaseType=dbo.tblDatabaseType.Databas
eType INNER JOIN
        dbo.tblMishapClass ON
dbo.tblMishaps.Class_FK=dbo.tblMishapClass.MishapClass
Code INNER JOIN
        dbo.tblMishapLocation ON
dbo.tblMishaps.LocationID_FK=dbo.tblMishapLocation.Mis
hapLocationID INNER JOIN
        dbo.tblMishapType ON
dbo.tblMishaps.Type_FK=dbo.tblMishapType.MishapTypeC
ode INNER JOIN
        dbo.tblOrganization ON
dbo.tblMishaps.OrgID_FK=dbo.tblOrganization .OrgID
```

```
WHERE  dbo.tblMishaps.Aircraft_FK =
COALESCE(@AC, dbo.tblMishaps.Aircraft_FK) AND
        dbo.tblMishaps.Type_FK = COALESCE(@Type,
dbo.tblMishaps.Type_FK) AND
        dbo.tblMishaps.Class_FK =
COALESCE(@Class, dbo.tblMishaps.Class_FK) AND
        dbo.tblMishaps.LocationID_FK =
COALESCE(@Loc, dbo.tblMishaps.LocationID_FK) AND
        dbo.tblMishaps.OrgID_FK = COALESCE(@Svc,
dbo.tblMishaps.OrgID_FK) AND
        datepart(year, dbo.tblMishaps.MishapDate) =
COALESCE(@Yr,
datepart(year, dbo.tblMishaps.MishapDate)) AND
```

```
dbo.tblMishaps.DatabaseType=dbo.tblDatabaseType.Databas
eType
```

```
return
```

### **4-0-1-0-flanCrossTabForGraphing**

Alter Procedure [4-0-1-0-flanCrossTabForGraphing]

```
(
    @colLeft varchar(500),
    @colTop varchar(500)
)

As

/* set nocount on */

execute dbo.rac @grpcol=@colLeft, @pvtcol=@colTop, @transform='count(*)', @from ='dbo.vwFlanForGraphs',
@where="",
    @printagg='n',@grand_totals='n', @row_totals='n', @emptycell='0'

return
```

### **8-0-0-0-NelsonReportAllMishaps**

Alter Procedure [8-0-0-0-NelsonReportAllMishaps]

```
(
    @AC_Type          varchar(10) = NULL,
    @Mishap_Type       varchar(3) = NULL,
    @Mishap_Class      varchar(1) = NULL,
    @Location          varchar(25) = NULL,
    @Service           varchar(10) = NULL,
    @Year              int          = NULL,
    @1stLevel          varchar(5) = NULL,
    @2ndLevel          varchar(5) = NULL,
    @3rdLevel          varchar(5) = NULL
)

As

Set nocount on

--Insert filtered data into Temp Filter_Table

SELECT DISTINCT tblMishaps.MishapID INTO
#tblTemp_Filter_Table

FROM    tblMishaps INNER JOIN
        tblDatabaseType ON
tblMishaps.DatabaseType=tblDatabaseType.DatabaseType
INNER JOIN
        tblMishapFactors ON
tblMishaps.MishapID = tblMishapFactors.MishapID_FK
INNER JOIN
        tblFactors ON
tblMishapFactors.[3rdLevelCode_FK] =
tblFactors.[3rdLevelCode]

WHERE   tblMishaps.Aircraft_FK =
COALESCE(@AC_Type, tblMishaps.Aircraft_FK) AND
        tblMishaps.Type_FK =
COALESCE(@Mishap_Type, tblMishaps.Type_FK) AND
        tblMishaps.Class_FK =
COALESCE(@Mishap_Class, tblMishaps.Class_FK) AND
        tblMishaps.LocationID_FK =
COALESCE(@Location, tblMishaps.LocationID_FK) AND
        tblMishaps.OrgID_FK = COALESCE(@Service,
tblMishaps.OrgID_FK) AND
        datepart(year, tblMishaps.MishapDate) =
COALESCE(@Year, datepart(year, tblMishaps.MishapDate))
AND
        tblFactors.[1stLevelCode] =
COALESCE(@1stLevel, tblFactors.[1stLevelCode]) AND
        tblFactors.[2ndLevelCode] =
COALESCE(@2ndLevel, tblFactors.[2ndLevelCode]) AND
        tblFactors.[3rdLevelCode] =
COALESCE(@3rdLevel, tblFactors.[3rdLevelCode]) AND

tblMishaps.DatabaseType=tblDatabaseType.DatabaseType

-----Build MishapCount resultset -----
SELECT
(
    SELECT Count([MishapID])
    FROM #tblTemp_Filter_Table
    WHERE (((#tblTemp_Filter_Table.MishapID) In (
        SELECT DISTINCT
        #tblTemp_Filter_Table.MishapID
        FROM #tblTemp_Filter_Table,
        tblFactors, tblMishapFactors
        WHERE
        #tblTemp_Filter_Table.MishapID =
        tblMishapFactors.MishapID_FK AND
        = tblMishapFactors.[3rdLevelCode_FK] AND (
        tblFactors.[1stLevelCode]
```

```
WHERE
#tblTemp_Filter_Table.MishapID =
tblMishapFactors.MishapID_FK AND
        tblFactors.[3rdLevelCode]
= tblMishapFactors.[3rdLevelCode_FK] AND (
        tblFactors.[1stLevelCode]
= 'MG')))) AS MG,
(
    SELECT Count([MishapID])
    FROM #tblTemp_Filter_Table
    WHERE (((#tblTemp_Filter_Table.MishapID) In (
        SELECT DISTINCT
        #tblTemp_Filter_Table.MishapID
        FROM #tblTemp_Filter_Table,
        tblFactors, tblMishapFactors
        WHERE
        #tblTemp_Filter_Table.MishapID =
        tblMishapFactors.MishapID_FK AND
        = tblMishapFactors.[3rdLevelCode_FK] AND (
        tblFactors.[1stLevelCode]
= 'MC')))) AS MC,
(
    SELECT Count([MishapID])
    FROM #tblTemp_Filter_Table
    WHERE (((#tblTemp_Filter_Table.MishapID) In (
        SELECT DISTINCT
        #tblTemp_Filter_Table.MishapID
        FROM #tblTemp_Filter_Table,
        tblFactors, tblMishapFactors
        WHERE
        #tblTemp_Filter_Table.MishapID =
        tblMishapFactors.MishapID_FK AND
        = tblMishapFactors.[3rdLevelCode_FK] AND (
        tblFactors.[1stLevelCode]
= 'WC')))) AS WC,
(
    SELECT Count([MishapID])
    FROM #tblTemp_Filter_Table
    WHERE (((#tblTemp_Filter_Table.MishapID) In (
        SELECT DISTINCT
        #tblTemp_Filter_Table.MishapID
        FROM #tblTemp_Filter_Table,
        tblFactors, tblMishapFactors
        WHERE
        #tblTemp_Filter_Table.MishapID =
        tblMishapFactors.MishapID_FK AND
        = tblMishapFactors.[3rdLevelCode_FK] AND (
        tblFactors.[1stLevelCode]
= 'MA')))) AS MA,
(
    SELECT Count([MishapID])
    FROM #tblTemp_Filter_Table
    WHERE (((#tblTemp_Filter_Table.MishapID) In (
        SELECT DISTINCT
        #tblTemp_Filter_Table.MishapID
        FROM #tblTemp_Filter_Table,
        tblFactors, tblMishapFactors
        WHERE
        #tblTemp_Filter_Table.MishapID =
        tblMishapFactors.MishapID_FK AND
        = tblMishapFactors.[3rdLevelCode_FK] AND (
```



```

tblFactors.[2ndLevelCode]
= 'VIO')))) AS VIO,
(
SELECT Count([MisapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MisapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MisapID
FROM #tblTemp_Filter_Table,
tblMisapFactors
WHERE
#tblTemp_Filter_Table.MisapID =
tblMisapFactors.MisapID_FK AND (

tblMisapFactors.[3rdLevelCode_FK] =
'PRO')))) AS PRO,
(
SELECT Count([MisapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MisapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MisapID
FROM #tblTemp_Filter_Table,
tblMisapFactors
WHERE
#tblTemp_Filter_Table.MisapID =
tblMisapFactors.MisapID_FK AND (

tblMisapFactors.[3rdLevelCode_FK] =
'DOC')))) AS DOC,
(
SELECT Count([MisapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MisapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MisapID
FROM #tblTemp_Filter_Table,
tblMisapFactors
WHERE
#tblTemp_Filter_Table.MisapID =
tblMisapFactors.MisapID_FK AND (

tblMisapFactors.[3rdLevelCode_FK] =
'DES')))) AS DES,
(
SELECT Count([MisapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MisapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MisapID
FROM #tblTemp_Filter_Table,
tblMisapFactors
WHERE
#tblTemp_Filter_Table.MisapID =
tblMisapFactors.MisapID_FK AND (

tblMisapFactors.[3rdLevelCode_FK] =
'RES')))) AS RES,
(
SELECT Count([MisapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MisapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MisapID
FROM #tblTemp_Filter_Table,
tblMisapFactors

```

```

WHERE
#tblTemp_Filter_Table.MisapID =
tblMisapFactors.MisapID_FK AND (

tblMisapFactors.[3rdLevelCode_FK] =
'IDQ')))) AS IDQ,
(
SELECT Count([MisapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MisapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MisapID
FROM #tblTemp_Filter_Table,
tblMisapFactors
WHERE
#tblTemp_Filter_Table.MisapID =
tblMisapFactors.MisapID_FK AND (

tblMisapFactors.[3rdLevelCode_FK] =
'OPS')))) AS OPS,
(
SELECT Count([MisapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MisapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MisapID
FROM #tblTemp_Filter_Table,
tblMisapFactors
WHERE
#tblTemp_Filter_Table.MisapID =
tblMisapFactors.MisapID_FK AND (

tblMisapFactors.[3rdLevelCode_FK] =
'PRB')))) AS PRB,
(
SELECT Count([MisapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MisapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MisapID
FROM #tblTemp_Filter_Table,
tblMisapFactors
WHERE
#tblTemp_Filter_Table.MisapID =
tblMisapFactors.MisapID_FK AND (

tblMisapFactors.[3rdLevelCode_FK] =
'MIS')))) AS MIS,
(
SELECT Count([MisapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MisapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MisapID
FROM #tblTemp_Filter_Table,
tblMisapFactors
WHERE
#tblTemp_Filter_Table.MisapID =
tblMisapFactors.MisapID_FK AND (

tblMisapFactors.[3rdLevelCode_FK] =
'MNT')))) AS MNT,
(
SELECT Count([MisapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MisapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MisapID

```







```

tblMishapFactors.[3rdLevelCode_FK] =
'KNW')')))) AS KNW,
(
SELECT Count([MishapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MishapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MishapID
FROM #tblTemp_Filter_Table,
tblMishapFactors
WHERE
#tblTemp_Filter_Table.MishapID =
tblMishapFactors.MishapID_FK AND (

tblMishapFactors.[3rdLevelCode_FK] =
'SKL')')))) AS SKL,
(
SELECT Count([MishapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MishapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MishapID
FROM #tblTemp_Filter_Table,
tblMishapFactors
WHERE
#tblTemp_Filter_Table.MishapID =
tblMishapFactors.MishapID_FK AND (

tblMishapFactors.[3rdLevelCode_FK] =
'ROU')')))) AS ROU,
(
SELECT Count([MishapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MishapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MishapID
FROM #tblTemp_Filter_Table,
tblMishapFactors
WHERE
#tblTemp_Filter_Table.MishapID =
tblMishapFactors.MishapID_FK AND (

```

```

tblMishapFactors.[3rdLevelCode_FK] = 'IFC')'))))
AS IFC,
(
SELECT Count([MishapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MishapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MishapID
FROM #tblTemp_Filter_Table,
tblMishapFactors
WHERE
#tblTemp_Filter_Table.MishapID =
tblMishapFactors.MishapID_FK AND (

tblMishapFactors.[3rdLevelCode_FK] =
'FLG')')))) AS FLG,
(
SELECT Count([MishapID])
FROM #tblTemp_Filter_Table
WHERE (((#tblTemp_Filter_Table.MishapID) In (
SELECT DISTINCT
#tblTemp_Filter_Table.MishapID
FROM #tblTemp_Filter_Table,
tblMishapFactors
WHERE
#tblTemp_Filter_Table.MishapID =
tblMishapFactors.MishapID_FK AND (

tblMishapFactors.[3rdLevelCode_FK] =
'EXC')')))) AS EXC,
(
SELECT Count([#tblTemp_Filter_Table].[MishapID])
FROM #tblTemp_Filter_Table)
AS TotalMishaps;

return

```

### 8-0-0-1-FlanReportByAircraft

Alter Procedure [8-0-0-1-FlanReportByAircraft]

As

SET NOCOUNT ON

```
CREATE TABLE #nResult3 (  
    Aircraft_FK varchar(255),  
    ADA int DEFAULT 0,  
    ASS int DEFAULT 0,  
    ATT int DEFAULT 0,  
    COM int DEFAULT 0,  
    CON int DEFAULT 0,  
    CRT int DEFAULT 0,  
    DES int DEFAULT 0,  
    DMG int DEFAULT 0,  
    DOC int DEFAULT 0,  
    DUC int DEFAULT 0,  
    EHZ int DEFAULT 0,  
    EXC int DEFAULT 0,  
    FLG int DEFAULT 0,  
    IDQ int DEFAULT 0,  
    IFC int DEFAULT 0,  
    INA int DEFAULT 0,  
    INF int DEFAULT 0,  
    JDG int DEFAULT 0,  
    KNW int DEFAULT 0,  
    LGT int DEFAULT 0,  
    LIM int DEFAULT 0,  
    MIS int DEFAULT 0,  
    MNT int DEFAULT 0,  
    OBS int DEFAULT 0,  
    OPS int DEFAULT 0,  
    PHY int DEFAULT 0,  
    PRB int DEFAULT 0,  
    PRO int DEFAULT 0,  
    RES int DEFAULT 0,  
    ROU int DEFAULT 0,  
    SKL int DEFAULT 0,  
    TRG int DEFAULT 0,  
    UNA int DEFAULT 0,  
    UNK int DEFAULT 0,  
    WXE int DEFAULT 0  
)
```

```
CREATE TABLE #nResult2 (  
    Aircraft_FK varchar(255),  
    CRW int DEFAULT 0,  
    ENV int DEFAULT 0,  
    EQP int DEFAULT 0,  
    ERR int DEFAULT 0,  
    MED int DEFAULT 0,  
    ORG int DEFAULT 0,  
    RDY int DEFAULT 0,  
    SUP int DEFAULT 0,  
    UNK int DEFAULT 0,  
    VIO int DEFAULT 0,  
    WRK int DEFAULT 0  
)
```

```
CREATE TABLE #nResult1 (  
    Aircraft_FK varchar(255),  
    MA int DEFAULT 0,  
    MC int DEFAULT 0,  
    MG int DEFAULT 0,  
    UN int DEFAULT 0,  
    WC int DEFAULT 0  
)
```

```
)  
CREATE TABLE #nResultFinal (  
    Aircraft_FK varchar(255),  
    ADA int DEFAULT 0,  
    ASS int DEFAULT 0,  
    ATT int DEFAULT 0,  
    COM int DEFAULT 0,  
    CON int DEFAULT 0,  
    CRT int DEFAULT 0,  
    DES int DEFAULT 0,  
    DMG int DEFAULT 0,  
    DOC int DEFAULT 0,  
    DUC int DEFAULT 0,  
    EHZ int DEFAULT 0,  
    EXC int DEFAULT 0,  
    FLG int DEFAULT 0,  
    IDQ int DEFAULT 0,  
    IFC int DEFAULT 0,  
    INA int DEFAULT 0,  
    INF int DEFAULT 0,  
    JDG int DEFAULT 0,  
    KNW int DEFAULT 0,  
    LIM int DEFAULT 0,  
    LGT int DEFAULT 0,  
    MIS int DEFAULT 0,  
    MNT int DEFAULT 0,  
    OBS int DEFAULT 0,  
    OPS int DEFAULT 0,  
    PHY int DEFAULT 0,  
    PRB int DEFAULT 0,  
    PRO int DEFAULT 0,  
    RES int DEFAULT 0,  
    ROU int DEFAULT 0,  
    SKL int DEFAULT 0,  
    TRG int DEFAULT 0,  
    UNA int DEFAULT 0,  
    WXE int DEFAULT 0,  
    CRW int DEFAULT 0,  
    WRK int DEFAULT 0,  
    ENV int DEFAULT 0,  
    EQP int DEFAULT 0,  
    ERR int DEFAULT 0,  
    MED int DEFAULT 0,  
    ORG int DEFAULT 0,  
    RDY int DEFAULT 0,  
    SUP int DEFAULT 0,  
    VIO int DEFAULT 0,  
    MA int DEFAULT 0,  
    MC int DEFAULT 0,  
    MG int DEFAULT 0,  
    WC int DEFAULT 0  
)
```

-----FOR THIRD LEVEL FACTORS

--Build a temp table and update the null values to 'None'

```
SELECT MishapID, [3rdLevelCode], Aircraft_FK INTO  
#nTemp3
```

```
FROM [vwFlanReports-2-2-Aircraft3]
```

```
UPDATE #nTemp3
```

```
SET Aircraft_FK = 'None'
```

```
WHERE Aircraft_FK is null
```

```
--Now run the crosstab
```

```
INSERT #nResult3
```

```

EXEC dbo.rac @grpcol='Aircraft_FK',
@pvtcol='[3rdLevelCode]', @transform='count(*)', @from
=#nTemp3', @where=',
    @printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'

----- FOR SECOND LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [2ndLevelCode], Aircraft_FK INTO
#nTemp2
FROM [vwFlanReports-2-2-Aircraft2]

UPDATE #nTemp2
SET Aircraft_FK = 'None'
WHERE Aircraft_FK is null
--Now run the crosstab
INSERT #nResult2
EXEC dbo.rac @grpcol='Aircraft_FK',
@pvtcol='[2ndLevelCode]', @transform='count(*)', @from
=#nTemp2', @where=',
    @printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'

-----FOR FIRST LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [1stLevelCode], Aircraft_FK INTO
#nTemp1
FROM [vwFlanReports-2-2-Aircraft1]

UPDATE #nTemp1
SET Aircraft_FK = 'None'
WHERE Aircraft_FK is null
--Now run the crosstab
INSERT #nResult1
EXEC dbo.rac @grpcol='Aircraft_FK',
@pvtcol='[1stLevelCode]', @transform='count(*)', @from
=#nTemp1', @where=',
    @printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'
-----

INSERT #nResultFinal
SELECT dbo.#nResult3.Aircraft_FK,
dbo.#nResult3.ADA, dbo.#nResult3.ASS,
dbo.#nResult3.ATT, dbo.#nResult3.COM,
dbo.#nResult3.CON, dbo.#nResult3.CRT,
dbo.#nResult3.DES,
    dbo.#nResult3.DMG,
dbo.#nResult3.DOC, dbo.#nResult3.DUC,
dbo.#nResult3.EHZ, dbo.#nResult3.EXC,
dbo.#nResult3.FLG, dbo.#nResult3.IDQ, dbo.#nResult3.IFC,
dbo.#nResult3.INA, dbo.#nResult3.INF,
    dbo.#nResult3.JDG,
dbo.#nResult3.KNW, dbo.#nResult3.LIM,
dbo.#nResult3.LGT, dbo.#nResult3.MIS,
dbo.#nResult3.MNT, dbo.#nResult3.OBS,
    dbo.#nResult3.OPS,
dbo.#nResult3.PHY, dbo.#nResult3.PRB,
dbo.#nResult3.PRO, dbo.#nResult3.RES,
dbo.#nResult3.ROU, dbo.#nResult3.SKL,
    dbo.#nResult3.TRG,
dbo.#nResult3.UNA, dbo.#nResult3.WXE,
dbo.#nResult3.CRW, dbo.#nResult2.WRK,
dbo.#nResult2.ENV, dbo.#nResult2.EQP,
dbo.#nResult2.ERR,

```

```

    dbo.#nResult2.MED,
dbo.#nResult2.ORG, dbo.#nResult2.RDY,
dbo.#nResult2.SUP, dbo.#nResult2.VIO, dbo.#nResult1.MA,
dbo.#nResult1.MC,
    dbo.#nResult1.MG,
dbo.#nResult1.WC
FROM dbo.#nResult3 INNER JOIN
    dbo.#nResult2 ON dbo.#nResult3.Aircraft_FK
= dbo.#nResult2.Aircraft_FK INNER JOIN
    dbo.#nResult1 ON dbo.#nResult3.Aircraft_FK
= dbo.#nResult1.Aircraft_FK

SELECT tblMishaps.Aircraft_FK,
Count(tblMishaps.MishapID) AS TotalMishaps INTO
#nResultTotal
FROM dbo.tblMishaps
GROUP BY tblMishaps.Aircraft_FK

SELECT dbo.#nResultFinal.Aircraft_FK,
dbo.#nResultFinal.ADA, dbo.#nResultFinal.ASS,
dbo.#nResultFinal.ATT, dbo.#nResultFinal.COM,
dbo.#nResultFinal.CON, dbo.#nResultFinal.CRT,
dbo.#nResultFinal.DES,
    dbo.#nResultFinal.DMG,
dbo.#nResultFinal.DOC, dbo.#nResultFinal.DUC,
dbo.#nResultFinal.EHZ, dbo.#nResultFinal.EXC,
dbo.#nResultFinal.FLG, dbo.#nResultFinal.IDQ,
dbo.#nResultFinal.IFC, dbo.#nResultFinal.INA,
dbo.#nResultFinal.INF,
    dbo.#nResultFinal.JDG,
dbo.#nResultFinal.KNW, dbo.#nResultFinal.LIM,
dbo.#nResultFinal.LGT, dbo.#nResultFinal.MIS,
dbo.#nResultFinal.MNT, dbo.#nResultFinal.OBS,
    dbo.#nResultFinal.OPS,
dbo.#nResultFinal.PHY, dbo.#nResultFinal.PRB,
dbo.#nResultFinal.PRO, dbo.#nResultFinal.RES,
dbo.#nResultFinal.ROU, dbo.#nResultFinal.SKL,
    dbo.#nResultFinal.TRG,
dbo.#nResultFinal.UNA, dbo.#nResultFinal.WXE,
dbo.#nResultFinal.CRW, dbo.#nResultFinal.WRK,
dbo.#nResultFinal.ENV, dbo.#nResultFinal.EQP,
dbo.#nResultFinal.ERR,
    dbo.#nResultFinal.MED,
dbo.#nResultFinal.ORG, dbo.#nResultFinal.RDY,
dbo.#nResultFinal.SUP, dbo.#nResultFinal.VIO,
dbo.#nResultFinal.MA, dbo.#nResultFinal.MC,
    dbo.#nResultFinal.MG,
dbo.#nResultFinal.WC,
    dbo.#nResultTotal.TotalMishaps
FROM dbo.#nResultFinal INNER JOIN
    dbo.#nResultTotal ON
dbl.#nResultFinal.Aircraft_FK=dbo.#nResultTotal.Aircraft_
FK

DROP TABLE #nResultFinal
DROP TABLE #nResultTotal
DROP TABLE #nResult3
DROP TABLE #nResult2
DROP TABLE #nResult1

```

return

### **8-0-0-2-NelsonReportByLocation**

Alter Procedure [8-0-0-2-NelsonReportByLocation]

As

SET NOCOUNT ON

```
CREATE TABLE #nResult3 (
    LocationID_FK varchar(255),
    ADA int DEFAULT 0,
    ASS int DEFAULT 0,
    ATT int DEFAULT 0,
    COM int DEFAULT 0,
    CON int DEFAULT 0,
    CRT int DEFAULT 0,
    DES int DEFAULT 0,
    DMG int DEFAULT 0,
    DOC int DEFAULT 0,
    DUC int DEFAULT 0,
    EHZ int DEFAULT 0,
    EXC int DEFAULT 0,
    FLG int DEFAULT 0,
    IDQ int DEFAULT 0,
    IFC int DEFAULT 0,
    INA int DEFAULT 0,
    INF int DEFAULT 0,
    JDG int DEFAULT 0,
    KNW int DEFAULT 0,
    LGT int DEFAULT 0,
    LIM int DEFAULT 0,
    MIS int DEFAULT 0,
    MNT int DEFAULT 0,
    OBS int DEFAULT 0,
    OPS int DEFAULT 0,
    PHY int DEFAULT 0,
    PRB int DEFAULT 0,
    PRO int DEFAULT 0,
    RES int DEFAULT 0,
    ROU int DEFAULT 0,
    SKL int DEFAULT 0,
    TRG int DEFAULT 0,
    UNA int DEFAULT 0,
    UNK int DEFAULT 0,
    WXE int DEFAULT 0
)
```

```
CREATE TABLE #nResult2 (
    LocationID_FK varchar(255),
    CRW int DEFAULT 0,
    ENV int DEFAULT 0,
    EQP int DEFAULT 0,
    ERR int DEFAULT 0,
    MED int DEFAULT 0,
    ORG int DEFAULT 0,
    RDY int DEFAULT 0,
    SUP int DEFAULT 0,
    UNK int DEFAULT 0,
    VIO int DEFAULT 0,
    WRK int DEFAULT 0
)
```

```
CREATE TABLE #nResult1 (
    LocationID_FK varchar(255),
    MA int DEFAULT 0,
    MC int DEFAULT 0,
    MG int DEFAULT 0,
    UN int DEFAULT 0,
    WC int DEFAULT 0
)
```

```
)
CREATE TABLE #nResultFinal (
    LocationID_FK varchar(255),
    ADA int DEFAULT 0,
    ASS int DEFAULT 0,
    ATT int DEFAULT 0,
    COM int DEFAULT 0,
    CON int DEFAULT 0,
    CRT int DEFAULT 0,
    DES int DEFAULT 0,
    DMG int DEFAULT 0,
    DOC int DEFAULT 0,
    DUC int DEFAULT 0,
    EHZ int DEFAULT 0,
    EXC int DEFAULT 0,
    FLG int DEFAULT 0,
    IDQ int DEFAULT 0,
    IFC int DEFAULT 0,
    INA int DEFAULT 0,
    INF int DEFAULT 0,
    JDG int DEFAULT 0,
    KNW int DEFAULT 0,
    LIM int DEFAULT 0,
    LGT int DEFAULT 0,
    MIS int DEFAULT 0,
    MNT int DEFAULT 0,
    OBS int DEFAULT 0,
    OPS int DEFAULT 0,
    PHY int DEFAULT 0,
    PRB int DEFAULT 0,
    PRO int DEFAULT 0,
    RES int DEFAULT 0,
    ROU int DEFAULT 0,
    SKL int DEFAULT 0,
    TRG int DEFAULT 0,
    UNA int DEFAULT 0,
    WXE int DEFAULT 0,
    CRW int DEFAULT 0,
    WRK int DEFAULT 0,
    ENV int DEFAULT 0,
    EQP int DEFAULT 0,
    ERR int DEFAULT 0,
    MED int DEFAULT 0,
    ORG int DEFAULT 0,
    RDY int DEFAULT 0,
    SUP int DEFAULT 0,
    VIO int DEFAULT 0,
    MA int DEFAULT 0,
    MC int DEFAULT 0,
    MG int DEFAULT 0,
    WC int DEFAULT 0
)
```

```
-----FOR THIRD LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [3rdLevelCode], LocationID_FK INTO
#nTemp3
FROM [vwFlanReports-2-3-Location3]

UPDATE #nTemp3
SET LocationID_FK = 'None'
WHERE LocationID_FK is null
--Now run the crosstab
INSERT #nResult3
```

```

EXEC dbo.rac @grpcol= 'LocationID_FK',
@pvtcol='[3rdLevelCode]', @transform='count(*)', @from
=#nTemp3', @where=",
    @printagg='n',@grand_totals='n',
@row_totals='n', @emptycell='0'

----- FOR SECOND LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [2ndLevelCode], LocationID_FK
INTO #nTemp2
FROM [vwFlanReports-2-3-Location2]

UPDATE #nTemp2
SET LocationID_FK = 'None'
WHERE LocationID_FK is null
--Now run the crosstab
INSERT #nResult2
EXEC dbo.rac @grpcol= 'LocationID_FK',
@pvtcol='[2ndLevelCode]', @transform='count(*)', @from
=#nTemp2', @where=",
    @printagg='n',@grand_totals='n',
@row_totals='n', @emptycell='0'

-----FOR FIRST LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [1stLevelCode], LocationID_FK INTO
#nTemp1
FROM [vwFlanReports-2-3-Location1]

UPDATE #nTemp1
SET LocationID_FK = 'None'
WHERE LocationID_FK is null
--Now run the crosstab
INSERT #nResult1
EXEC dbo.rac @grpcol= 'LocationID_FK',
@pvtcol='[1stLevelCode]', @transform='count(*)', @from
=#nTemp1', @where=",
    @printagg='n',@grand_totals='n',
@row_totals='n', @emptycell='0'

-----

INSERT #nResultFinal
SELECT dbo.#nResult3.LocationID_FK,
dbo.#nResult3.ADA, dbo.#nResult3.ASS,
dbo.#nResult3.ATT, dbo.#nResult3.COM,
dbo.#nResult3.CON, dbo.#nResult3.CRT,
dbo.#nResult3.DES,
    dbo.#nResult3.DMG,
dbo.#nResult3.DOC, dbo.#nResult3.DUC,
dbo.#nResult3.EHZ, dbo.#nResult3.EXC,
dbo.#nResult3.FLG, dbo.#nResult3.IDQ, dbo.#nResult3.IFC,
dbo.#nResult3.INA, dbo.#nResult3.INF,
    dbo.#nResult3.JDG,
dbo.#nResult3.KNW, dbo.#nResult3.LIM,
dbo.#nResult3.LGT, dbo.#nResult3.MIS,
dbo.#nResult3.MNT, dbo.#nResult3.OBS,
    dbo.#nResult3.OPS,
dbo.#nResult3.PHY, dbo.#nResult3.PRB,
dbo.#nResult3.PRO, dbo.#nResult3.RES,
dbo.#nResult3.ROU, dbo.#nResult3.SKL,
    dbo.#nResult3.TRG,
dbo.#nResult3.UNA, dbo.#nResult3.WXE,
dbo.#nResult2.CRW, dbo.#nResult2.WRK,
dbo.#nResult2.ENV, dbo.#nResult2.EQP,
dbo.#nResult2.ERR,
    dbo.#nResult2.MED,
dbo.#nResult2.ORG, dbo.#nResult2.RDY,

```

```

dbo.#nResult2.SUP, dbo.#nResult2.VIO, dbo.#nResult1.MA,
dbo.#nResult1.MC,
    dbo.#nResult1.MG,
dbo.#nResult1.WC
FROM dbo.#nResult3 INNER JOIN
    dbo.#nResult2 ON
    dbo.#nResult3.LocationID_FK =
    dbo.#nResult2.LocationID_FK INNER JOIN
        dbo.#nResult1 ON
        dbo.#nResult3.LocationID_FK =
        dbo.#nResult1.LocationID_FK

SELECT tblMishaps.LocationID_FK,
Count(tblMishaps.MishapID) AS TotalMishaps INTO
#nResultTotal
FROM dbo.tblMishaps
GROUP BY tblMishaps.LocationID_FK

SELECT dbo.#nResultFinal.LocationID_FK,
dbo.#nResultFinal.ADA, dbo.#nResultFinal.ASS,
dbo.#nResultFinal.ATT, dbo.#nResultFinal.COM,
dbo.#nResultFinal.CON, dbo.#nResultFinal.CRT,
dbo.#nResultFinal.DES,
    dbo.#nResultFinal.DMG,
dbo.#nResultFinal.DOC, dbo.#nResultFinal.DUC,
dbo.#nResultFinal.EHZ, dbo.#nResultFinal.EXC,
dbo.#nResultFinal.FLG, dbo.#nResultFinal.IDQ,
dbo.#nResultFinal.IFC, dbo.#nResultFinal.INA,
dbo.#nResultFinal.INF,
    dbo.#nResultFinal.JDG,
dbo.#nResultFinal.KNW, dbo.#nResultFinal.LIM,
dbo.#nResultFinal.LGT, dbo.#nResultFinal.MIS,
dbo.#nResultFinal.MNT, dbo.#nResultFinal.OBS,
    dbo.#nResultFinal.OPS,
dbo.#nResultFinal.PHY, dbo.#nResultFinal.PRB,
dbo.#nResultFinal.PRO, dbo.#nResultFinal.RES,
dbo.#nResultFinal.ROU, dbo.#nResultFinal.SKL,
    dbo.#nResultFinal.TRG,
dbo.#nResultFinal.UNA, dbo.#nResultFinal.WXE,
dbo.#nResultFinal.CRW, dbo.#nResultFinal.WRK,
dbo.#nResultFinal.ENV, dbo.#nResultFinal.EQP,
dbo.#nResultFinal.ERR,
    dbo.#nResultFinal.MED,
dbo.#nResultFinal.ORG, dbo.#nResultFinal.RDY,
dbo.#nResultFinal.SUP, dbo.#nResultFinal.VIO,
dbo.#nResultFinal.MA, dbo.#nResultFinal.MC,
    dbo.#nResultFinal.MG,
dbo.#nResultFinal.WC, dbo.#nResultTotal.TotalMishaps,
dbo.tblMishapLocation.MishapLocation

FROM dbo.#nResultFinal INNER JOIN
    dbo.#nResultTotal ON
    dbo.#nResultFinal.LocationID_FK =
    dbo.#nResultTotal.LocationID_FK INNER JOIN
        dbo.tblMishapLocation ON
        dbo.#nResultFinal.LocationID_FK =
        dbo.tblMishapLocation.MishapLocationID

DROP TABLE #nResultFinal
DROP TABLE #nResultTotal
DROP TABLE #nResult3
DROP TABLE #nResult2

```

```
DROP TABLE #nResult1  
return
```

### 8-0-0-3-NelsonReportByClass

Alter Procedure [8-0-0-3-NelsonReportByClass]

As

SET NOCOUNT ON

```
CREATE TABLE #nResult3 (  
    Class_FK varchar(255),  
    ADA int DEFAULT 0,  
    ASS int DEFAULT 0,  
    ATT int DEFAULT 0,  
    COM int DEFAULT 0,  
    CON int DEFAULT 0,  
    CRT int DEFAULT 0,  
    DES int DEFAULT 0,  
    DMG int DEFAULT 0,  
    DOC int DEFAULT 0,  
    DUC int DEFAULT 0,  
    EHZ int DEFAULT 0,  
    EXC int DEFAULT 0,  
    FLG int DEFAULT 0,  
    IDQ int DEFAULT 0,  
    IFC int DEFAULT 0,  
    INA int DEFAULT 0,  
    INF int DEFAULT 0,  
    JDG int DEFAULT 0,  
    KNW int DEFAULT 0,  
    LGT int DEFAULT 0,  
    LIM int DEFAULT 0,  
    MIS int DEFAULT 0,  
    MNT int DEFAULT 0,  
    OBS int DEFAULT 0,  
    OPS int DEFAULT 0,  
    PHY int DEFAULT 0,  
    PRB int DEFAULT 0,  
    PRO int DEFAULT 0,  
    RES int DEFAULT 0,  
    ROU int DEFAULT 0,  
    SKL int DEFAULT 0,  
    TRG int DEFAULT 0,  
    UNA int DEFAULT 0,  
    UNK int DEFAULT 0,  
    WXE int DEFAULT 0  
)
```

```
CREATE TABLE #nResult2 (  
    Class_FK varchar(255),  
    CRW int DEFAULT 0,  
    ENV int DEFAULT 0,  
    EQP int DEFAULT 0,  
    ERR int DEFAULT 0,  
    MED int DEFAULT 0,  
    ORG int DEFAULT 0,  
    RDY int DEFAULT 0,  
    SUP int DEFAULT 0,  
    UNK int DEFAULT 0,  
    VIO int DEFAULT 0,  
    WRK int DEFAULT 0  
)
```

```
CREATE TABLE #nResult1 (  
    Class_FK varchar(255),  
    MA int DEFAULT 0,  
    MC int DEFAULT 0,  
    MG int DEFAULT 0,  
    UN int DEFAULT 0,  
    WC int DEFAULT 0  
)
```

```
)  
CREATE TABLE #nResultFinal (  
    Class_FK varchar(255),  
    ADA int DEFAULT 0,  
    ASS int DEFAULT 0,  
    ATT int DEFAULT 0,  
    COM int DEFAULT 0,  
    CON int DEFAULT 0,  
    CRT int DEFAULT 0,  
    DES int DEFAULT 0,  
    DMG int DEFAULT 0,  
    DOC int DEFAULT 0,  
    DUC int DEFAULT 0,  
    EHZ int DEFAULT 0,  
    EXC int DEFAULT 0,  
    FLG int DEFAULT 0,  
    IDQ int DEFAULT 0,  
    IFC int DEFAULT 0,  
    INA int DEFAULT 0,  
    INF int DEFAULT 0,  
    JDG int DEFAULT 0,  
    KNW int DEFAULT 0,  
    LIM int DEFAULT 0,  
    LGT int DEFAULT 0,  
    MIS int DEFAULT 0,  
    MNT int DEFAULT 0,  
    OBS int DEFAULT 0,  
    OPS int DEFAULT 0,  
    PHY int DEFAULT 0,  
    PRB int DEFAULT 0,  
    PRO int DEFAULT 0,  
    RES int DEFAULT 0,  
    ROU int DEFAULT 0,  
    SKL int DEFAULT 0,  
    TRG int DEFAULT 0,  
    UNA int DEFAULT 0,  
    WXE int DEFAULT 0,  
    CRW int DEFAULT 0,  
    WRK int DEFAULT 0,  
    ENV int DEFAULT 0,  
    EQP int DEFAULT 0,  
    ERR int DEFAULT 0,  
    MED int DEFAULT 0,  
    ORG int DEFAULT 0,  
    RDY int DEFAULT 0,  
    SUP int DEFAULT 0,  
    VIO int DEFAULT 0,  
    MA int DEFAULT 0,  
    MC int DEFAULT 0,  
    MG int DEFAULT 0,  
    WC int DEFAULT 0  
)
```

-----FOR THIRD LEVEL FACTORS

--Build a temp table and update the null values to 'None'

```
SELECT MishapID, [3rdLevelCode], Class_FK INTO  
#nTemp3
```

```
FROM [vwFlanReports-2-4-Class3]
```

```
UPDATE #nTemp3
```

```
SET Class_FK = 'None'
```

```
WHERE Class_FK is null
```

```
--Now run the crosstab
```

```
INSERT #nResult3
```



```

EXEC dbo.rac @grpcol='Class_FK',
@pvtcol='[3rdLevelCode]', @transform='count(*)', @from
=#nTemp3', @where=",
@printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'

----- FOR SECOND LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [2ndLevelCode], Class_FK INTO
#nTemp2
FROM [vwFlanReports-2-4-Class2]

UPDATE #nTemp2
SET Class_FK = 'None'
WHERE Class_FK is null
--Now run the crosstab
INSERT #nResult2
EXEC dbo.rac @grpcol='Class_FK',
@pvtcol='[2ndLevelCode]', @transform='count(*)', @from
=#nTemp2', @where=",
@printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'

----- FOR FIRST LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [1stLevelCode], Class_FK INTO
#nTemp1
FROM [vwFlanReports-2-4-Class1]

UPDATE #nTemp1
SET Class_FK = 'None'
WHERE Class_FK is null
--Now run the crosstab
INSERT #nResult1
EXEC dbo.rac @grpcol='Class_FK',
@pvtcol='[1stLevelCode]', @transform='count(*)', @from
=#nTemp1', @where=",
@printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'

-----

INSERT #nResultFinal
SELECT dbo.#nResult3.Class_FK,
dbo.#nResult3.ADA, dbo.#nResult3.ASS,
dbo.#nResult3.ATT, dbo.#nResult3.COM,
dbo.#nResult3.CON, dbo.#nResult3.CRT,
dbo.#nResult3.DES,
dbo.#nResult3.DMG,
dbo.#nResult3.DOC, dbo.#nResult3.DUC,
dbo.#nResult3.EHZ, dbo.#nResult3.EXC,
dbo.#nResult3.FLG, dbo.#nResult3.IDQ, dbo.#nResult3.IFC,
dbo.#nResult3.INA, dbo.#nResult3.INF,
dbo.#nResult3.JDG,
dbo.#nResult3.KNW, dbo.#nResult3.LIM,
dbo.#nResult3.LGT, dbo.#nResult3.MIS,
dbo.#nResult3.MNT, dbo.#nResult3.OBS,
dbo.#nResult3.OPS,
dbo.#nResult3.PHY, dbo.#nResult3.PRB,
dbo.#nResult3.PRO, dbo.#nResult3.RES,
dbo.#nResult3.ROU, dbo.#nResult3.SKL,
dbo.#nResult3.TRG,
dbo.#nResult3.UNA, dbo.#nResult3.WXE,
dbo.#nResult3.CRW, dbo.#nResult2.WRK,
dbo.#nResult2.ENV, dbo.#nResult2.EQP,
dbo.#nResult2.ERR,

```

```

dbo.#nResult2.MED,
dbo.#nResult2.ORG, dbo.#nResult2.RDY,
dbo.#nResult2.SUP, dbo.#nResult2.VIO, dbo.#nResult1.MA,
dbo.#nResult1.MC,
dbo.#nResult1.MG,
dbo.#nResult1.WC
FROM dbo.#nResult3 INNER JOIN
dbo.#nResult2 ON dbo.#nResult3.Class_FK =
dbo.#nResult2.Class_FK INNER JOIN
dbo.#nResult1 ON dbo.#nResult3.Class_FK =
dbo.#nResult1.Class_FK

SELECT tblMishaps.Class_FK,
Count(tblMishaps.MishapID) AS TotalMishaps INTO
#nResultTotal
FROM dbo.tblMishaps
GROUP BY tblMishaps.Class_FK

SELECT dbo.#nResultFinal.Class_FK,
dbo.#nResultFinal.ADA, dbo.#nResultFinal.ASS,
dbo.#nResultFinal.ATT, dbo.#nResultFinal.COM,
dbo.#nResultFinal.CON, dbo.#nResultFinal.CRT,
dbo.#nResultFinal.DES,
dbo.#nResultFinal.DMG,
dbo.#nResultFinal.DOC, dbo.#nResultFinal.DUC,
dbo.#nResultFinal.EHZ, dbo.#nResultFinal.EXC,
dbo.#nResultFinal.FLG, dbo.#nResultFinal.IDQ,
dbo.#nResultFinal.IFC, dbo.#nResultFinal.INA,
dbo.#nResultFinal.INF,
dbo.#nResultFinal.JDG,
dbo.#nResultFinal.KNW, dbo.#nResultFinal.LIM,
dbo.#nResultFinal.LGT, dbo.#nResultFinal.MIS,
dbo.#nResultFinal.MNT, dbo.#nResultFinal.OBS,
dbo.#nResultFinal.OPS,
dbo.#nResultFinal.PHY, dbo.#nResultFinal.PRB,
dbo.#nResultFinal.PRO, dbo.#nResultFinal.RES,
dbo.#nResultFinal.ROU, dbo.#nResultFinal.SKL,
dbo.#nResultFinal.TRG,
dbo.#nResultFinal.UNA, dbo.#nResultFinal.WXE,
dbo.#nResultFinal.CRW, dbo.#nResultFinal.WRK,
dbo.#nResultFinal.ENV, dbo.#nResultFinal.EQP,
dbo.#nResultFinal.ERR,
dbo.#nResultFinal.MED,
dbo.#nResultFinal.ORG, dbo.#nResultFinal.RDY,
dbo.#nResultFinal.SUP, dbo.#nResultFinal.VIO,
dbo.#nResultFinal.MA, dbo.#nResultFinal.MC,
dbo.#nResultFinal.MG,
dbo.#nResultFinal.WC,
dbo.#nResultTotal.TotalMishaps
FROM dbo.#nResultFinal INNER JOIN
dbo.#nResultTotal ON
dbl.#nResultFinal.Class_FK=dbo.#nResultTotal.Class_FK

DROP TABLE #nResultFinal
DROP TABLE #nResultTotal
DROP TABLE #nResult3
DROP TABLE #nResult2
DROP TABLE #nResult1

```

return

### 8-0-0-4-NelsonReportByOrganization

Alter Procedure [8-0-0-4-NelsonReportByOrganization]

As

SET NOCOUNT ON

```
CREATE TABLE #nResult3 (  
    OrgID_FK varchar(255),  
    ADA int DEFAULT 0,  
    ASS int DEFAULT 0,  
    ATT int DEFAULT 0,  
    COM int DEFAULT 0,  
    CON int DEFAULT 0,  
    CRT int DEFAULT 0,  
    DES int DEFAULT 0,  
    DMG int DEFAULT 0,  
    DOC int DEFAULT 0,  
    DUC int DEFAULT 0,  
    EHZ int DEFAULT 0,  
    EXC int DEFAULT 0,  
    FLG int DEFAULT 0,  
    IDQ int DEFAULT 0,  
    IFC int DEFAULT 0,  
    INA int DEFAULT 0,  
    INF int DEFAULT 0,  
    JDG int DEFAULT 0,  
    KNW int DEFAULT 0,  
    LGT int DEFAULT 0,  
    LIM int DEFAULT 0,  
    MIS int DEFAULT 0,  
    MNT int DEFAULT 0,  
    OBS int DEFAULT 0,  
    OPS int DEFAULT 0,  
    PHY int DEFAULT 0,  
    PRB int DEFAULT 0,  
    PRO int DEFAULT 0,  
    RES int DEFAULT 0,  
    ROU int DEFAULT 0,  
    SKL int DEFAULT 0,  
    TRG int DEFAULT 0,  
    UNA int DEFAULT 0,  
    UNK int DEFAULT 0,  
    WXE int DEFAULT 0  
)
```

```
CREATE TABLE #nResult2 (  
    OrgID_FK varchar(255),  
    CRW int DEFAULT 0,  
    ENV int DEFAULT 0,  
    EQP int DEFAULT 0,  
    ERR int DEFAULT 0,  
    MED int DEFAULT 0,  
    ORG int DEFAULT 0,  
    RDY int DEFAULT 0,  
    SUP int DEFAULT 0,  
    UNK int DEFAULT 0,  
    VIO int DEFAULT 0,  
    WRK int DEFAULT 0  
)
```

```
CREATE TABLE #nResult1 (  
    OrgID_FK varchar(255),  
    MA int DEFAULT 0,  
    MC int DEFAULT 0,  
    MG int DEFAULT 0,  
    UN int DEFAULT 0,  
    WC int DEFAULT 0  
)
```

```
)  
CREATE TABLE #nResultFinal (  
    OrgID_FK varchar(255),  
    ADA int DEFAULT 0,  
    ASS int DEFAULT 0,  
    ATT int DEFAULT 0,  
    COM int DEFAULT 0,  
    CON int DEFAULT 0,  
    CRT int DEFAULT 0,  
    DES int DEFAULT 0,  
    DMG int DEFAULT 0,  
    DOC int DEFAULT 0,  
    DUC int DEFAULT 0,  
    EHZ int DEFAULT 0,  
    EXC int DEFAULT 0,  
    FLG int DEFAULT 0,  
    IDQ int DEFAULT 0,  
    IFC int DEFAULT 0,  
    INA int DEFAULT 0,  
    INF int DEFAULT 0,  
    JDG int DEFAULT 0,  
    KNW int DEFAULT 0,  
    LIM int DEFAULT 0,  
    LGT int DEFAULT 0,  
    MIS int DEFAULT 0,  
    MNT int DEFAULT 0,  
    OBS int DEFAULT 0,  
    OPS int DEFAULT 0,  
    PHY int DEFAULT 0,  
    PRB int DEFAULT 0,  
    PRO int DEFAULT 0,  
    RES int DEFAULT 0,  
    ROU int DEFAULT 0,  
    SKL int DEFAULT 0,  
    TRG int DEFAULT 0,  
    UNA int DEFAULT 0,  
    WXE int DEFAULT 0,  
    CRW int DEFAULT 0,  
    WRK int DEFAULT 0,  
    ENV int DEFAULT 0,  
    EQP int DEFAULT 0,  
    ERR int DEFAULT 0,  
    MED int DEFAULT 0,  
    ORG int DEFAULT 0,  
    RDY int DEFAULT 0,  
    SUP int DEFAULT 0,  
    VIO int DEFAULT 0,  
    MA int DEFAULT 0,  
    MC int DEFAULT 0,  
    MG int DEFAULT 0,  
    WC int DEFAULT 0  
)
```

-----FOR THIRD LEVEL FACTORS

--Build a temp table and update the null values to 'None'

```
SELECT MishapID, [3rdLevelCode], OrgID_FK INTO  
#nTemp3
```

```
FROM [vwFlanReports-2-5-Organization3]
```

```
UPDATE #nTemp3
```

```
SET OrgID_FK = 'None'
```

```
WHERE OrgID_FK is null
```

```
--Now run the crosstab
```

```
INSERT #nResult3
```

```

EXEC dbo.rac @grpcol='OrgID_FK ',
@pvtcol='[3rdLevelCode]', @transform='count(*)', @from
=#nTemp3', @where=",
@printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'

----- FOR SECOND LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [2ndLevelCode], OrgID_FK INTO
#nTemp2
FROM [vwFlanReports-2-5-Organization2]

UPDATE #nTemp2
SET OrgID_FK = 'None'
WHERE OrgID_FK is null
--Now run the crosstab
INSERT #nResult2
EXEC dbo.rac @grpcol='OrgID_FK ',
@pvtcol='[2ndLevelCode]', @transform='count(*)', @from
=#nTemp2', @where=",
@printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'

----- FOR FIRST LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [1stLevelCode], OrgID_FK INTO
#nTemp1
FROM [vwFlanReports-2-5-Organization1]

UPDATE #nTemp1
SET OrgID_FK = 'None'
WHERE OrgID_FK is null
--Now run the crosstab
INSERT #nResult1
EXEC dbo.rac @grpcol='OrgID_FK ',
@pvtcol='[1stLevelCode]', @transform='count(*)', @from
=#nTemp1', @where=",
@printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'

-----
INSERT #nResultFinal
SELECT dbo.#nResult3.OrgID_FK ,
dbo.#nResult3.ADA, dbo.#nResult3.ASS,
dbo.#nResult3.ATT, dbo.#nResult3.COM,
dbo.#nResult3.CON, dbo.#nResult3.CRT,
dbo.#nResult3.DES,
dbo.#nResult3.DMG,
dbo.#nResult3.DOC, dbo.#nResult3.DUC,
dbo.#nResult3.EHZ, dbo.#nResult3.EXC,
dbo.#nResult3.FLG, dbo.#nResult3.IDQ, dbo.#nResult3.IFC,
dbo.#nResult3.INA, dbo.#nResult3.INF,
dbo.#nResult3.JDG,
dbo.#nResult3.KNW, dbo.#nResult3.LIM,
dbo.#nResult3.LGT, dbo.#nResult3.MIS,
dbo.#nResult3.MNT, dbo.#nResult3.OBS,
dbo.#nResult3.OPS,
dbo.#nResult3.PHY, dbo.#nResult3.PRB,
dbo.#nResult3.PRO, dbo.#nResult3.RES,
dbo.#nResult3.ROU, dbo.#nResult3.SKL,
dbo.#nResult3.TRG,
dbo.#nResult3.UNA, dbo.#nResult3.WXE,
dbo.#nResult2.CRW, dbo.#nResult2.WRK,
dbo.#nResult2.ENV, dbo.#nResult2.EQP,
dbo.#nResult2.ERR,

```

```

dbo.#nResult2.MED,
dbo.#nResult2.ORG, dbo.#nResult2.RDY,
dbo.#nResult2.SUP, dbo.#nResult2.VIO, dbo.#nResult1.MA,
dbo.#nResult1.MC,
dbo.#nResult1.MG,
dbo.#nResult1.WC
FROM dbo.#nResult3 INNER JOIN
dbo.#nResult2 ON dbo.#nResult3.OrgID_FK
= dbo.#nResult2.OrgID_FK INNER JOIN
dbo.#nResult1 ON dbo.#nResult3.OrgID_FK
= dbo.#nResult1.OrgID_FK

SELECT tblMishaps.OrgID_FK ,
Count(tblMishaps.MishapID) AS TotalMishaps INTO
#nResultTotal
FROM dbo.tblMishaps
GROUP BY tblMishaps.OrgID_FK

SELECT dbo.#nResultFinal.OrgID_FK ,
dbo.#nResultFinal.ADA, dbo.#nResultFinal.ASS,
dbo.#nResultFinal.ATT, dbo.#nResultFinal.COM,
dbo.#nResultFinal.CON, dbo.#nResultFinal.CRT,
dbo.#nResultFinal.DES,
dbo.#nResultFinal.DMG,
dbo.#nResultFinal.DOC, dbo.#nResultFinal.DUC,
dbo.#nResultFinal.EHZ, dbo.#nResultFinal.EXC,
dbo.#nResultFinal.FLG, dbo.#nResultFinal.IDQ,
dbo.#nResultFinal.IFC, dbo.#nResultFinal.INA,
dbo.#nResultFinal.INF,
dbo.#nResultFinal.JDG,
dbo.#nResultFinal.KNW, dbo.#nResultFinal.LIM,
dbo.#nResultFinal.LGT, dbo.#nResultFinal.MIS,
dbo.#nResultFinal.MNT, dbo.#nResultFinal.OBS,
dbo.#nResultFinal.OPS,
dbo.#nResultFinal.PHY, dbo.#nResultFinal.PRB,
dbo.#nResultFinal.PRO, dbo.#nResultFinal.RES,
dbo.#nResultFinal.ROU, dbo.#nResultFinal.SKL,
dbo.#nResultFinal.TRG,
dbo.#nResultFinal.UNA, dbo.#nResultFinal.WXE,
dbo.#nResultFinal.CRW, dbo.#nResultFinal.WRK,
dbo.#nResultFinal.ENV, dbo.#nResultFinal.EQP,
dbo.#nResultFinal.ERR,
dbo.#nResultFinal.MED,
dbo.#nResultFinal.ORG, dbo.#nResultFinal.RDY,
dbo.#nResultFinal.SUP, dbo.#nResultFinal.VIO,
dbo.#nResultFinal.MA, dbo.#nResultFinal.MC,
dbo.#nResultFinal.MG,
dbo.#nResultFinal.WC,
dbo.#nResultTotal.TotalMishaps
FROM dbo.#nResultFinal INNER JOIN
dbo.#nResultTotal ON
dbl.#nResultFinal.OrgID_FK
=dbo.#nResultTotal.OrgID_FK

DROP TABLE #nResultFinal
DROP TABLE #nResultTotal
DROP TABLE #nResult3
DROP TABLE #nResult2
DROP TABLE #nResult1

```

return

### 8-0-0-5-NelsonReportByType

Alter Procedure [8-0-0-5-NelsonReportByType]

As

SET NOCOUNT ON

```
CREATE TABLE #nResult3 (  
    Type_FK varchar(255),  
    ADA int DEFAULT 0,  
    ASS int DEFAULT 0,  
    ATT int DEFAULT 0,  
    COM int DEFAULT 0,  
    CON int DEFAULT 0,  
    CRT int DEFAULT 0,  
    DES int DEFAULT 0,  
    DMG int DEFAULT 0,  
    DOC int DEFAULT 0,  
    DUC int DEFAULT 0,  
    EHZ int DEFAULT 0,  
    EXC int DEFAULT 0,  
    FLG int DEFAULT 0,  
    IDQ int DEFAULT 0,  
    IFC int DEFAULT 0,  
    INA int DEFAULT 0,  
    INF int DEFAULT 0,  
    JDG int DEFAULT 0,  
    KNW int DEFAULT 0,  
    LGT int DEFAULT 0,  
    LIM int DEFAULT 0,  
    MIS int DEFAULT 0,  
    MNT int DEFAULT 0,  
    OBS int DEFAULT 0,  
    OPS int DEFAULT 0,  
    PHY int DEFAULT 0,  
    PRB int DEFAULT 0,  
    PRO int DEFAULT 0,  
    RES int DEFAULT 0,  
    ROU int DEFAULT 0,  
    SKL int DEFAULT 0,  
    TRG int DEFAULT 0,  
    UNA int DEFAULT 0,  
    UNK int DEFAULT 0,  
    WXE int DEFAULT 0  
)
```

```
CREATE TABLE #nResult2 (  
    Type_FK varchar(255),  
    CRW int DEFAULT 0,  
    ENV int DEFAULT 0,  
    EQP int DEFAULT 0,  
    ERR int DEFAULT 0,  
    MED int DEFAULT 0,  
    ORG int DEFAULT 0,  
    RDY int DEFAULT 0,  
    SUP int DEFAULT 0,  
    UNK int DEFAULT 0,  
    VIO int DEFAULT 0,  
    WRK int DEFAULT 0  
)
```

```
CREATE TABLE #nResult1 (  
    Type_FK varchar(255),  
    MA int DEFAULT 0,  
    MC int DEFAULT 0,  
    MG int DEFAULT 0,  
    UN int DEFAULT 0,  
    WC int DEFAULT 0  
)
```

```
)  
CREATE TABLE #nResultFinal (  
    Type_FK varchar(255),  
    ADA int DEFAULT 0,  
    ASS int DEFAULT 0,  
    ATT int DEFAULT 0,  
    COM int DEFAULT 0,  
    CON int DEFAULT 0,  
    CRT int DEFAULT 0,  
    DES int DEFAULT 0,  
    DMG int DEFAULT 0,  
    DOC int DEFAULT 0,  
    DUC int DEFAULT 0,  
    EHZ int DEFAULT 0,  
    EXC int DEFAULT 0,  
    FLG int DEFAULT 0,  
    IDQ int DEFAULT 0,  
    IFC int DEFAULT 0,  
    INA int DEFAULT 0,  
    INF int DEFAULT 0,  
    JDG int DEFAULT 0,  
    KNW int DEFAULT 0,  
    LIM int DEFAULT 0,  
    LGT int DEFAULT 0,  
    MIS int DEFAULT 0,  
    MNT int DEFAULT 0,  
    OBS int DEFAULT 0,  
    OPS int DEFAULT 0,  
    PHY int DEFAULT 0,  
    PRB int DEFAULT 0,  
    PRO int DEFAULT 0,  
    RES int DEFAULT 0,  
    ROU int DEFAULT 0,  
    SKL int DEFAULT 0,  
    TRG int DEFAULT 0,  
    UNA int DEFAULT 0,  
    WXE int DEFAULT 0,  
    CRW int DEFAULT 0,  
    WRK int DEFAULT 0,  
    ENV int DEFAULT 0,  
    EQP int DEFAULT 0,  
    ERR int DEFAULT 0,  
    MED int DEFAULT 0,  
    ORG int DEFAULT 0,  
    RDY int DEFAULT 0,  
    SUP int DEFAULT 0,  
    VIO int DEFAULT 0,  
    MA int DEFAULT 0,  
    MC int DEFAULT 0,  
    MG int DEFAULT 0,  
    WC int DEFAULT 0  
)
```

```
-----  
-----FOR THIRD LEVEL FACTORS  
--Build a temp table and update the null values to 'None'  
SELECT MishapID, [3rdLevelCode], Type_FK INTO  
#nTemp3  
FROM [vwFlanReports-2-6-Type3]  
  
UPDATE #nTemp3  
SET Type_FK = 'None'  
WHERE Type_FK is null  
--Now run the crosstab  
INSERT #nResult3
```

```

EXEC dbo.rac @grpcol='Type_FK ',
@pvtcol='[3rdLevelCode]', @transform='count(*)', @from
=#nTemp3', @where=",
@printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'

----- FOR SECOND LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [2ndLevelCode], Type_FK INTO
#nTemp2
FROM [vwFlanReports-2-6-Type2]

UPDATE #nTemp2
SET Type_FK = 'None'
WHERE Type_FK is null
--Now run the crosstab
INSERT #nResult2
EXEC dbo.rac @grpcol='Type_FK ',
@pvtcol='[2ndLevelCode]', @transform='count(*)', @from
=#nTemp2', @where=",
@printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'

----- FOR FIRST LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [1stLevelCode], Type_FK INTO
#nTemp1
FROM [vwFlanReports-2-6-Type1]

UPDATE #nTemp1
SET Type_FK = 'None'
WHERE Type_FK is null
--Now run the crosstab
INSERT #nResult1
EXEC dbo.rac @grpcol='Type_FK ',
@pvtcol='[1stLevelCode]', @transform='count(*)', @from
=#nTemp1', @where=",
@printagg='n', @grand_totals='n',
@row_totals='n', @emptycell='0'

-----

INSERT #nResult Final
SELECT dbo.#nResult3.Type_FK ,
dbo.#nResult3.ADA, dbo.#nResult3.ASS,
dbo.#nResult3.ATT, dbo.#nResult3.COM,
dbo.#nResult3.CON, dbo.#nResult3.CRT,
dbo.#nResult3.DES,
dbo.#nResult3.DMG,
dbo.#nResult3.DOC, dbo.#nResult3.DUC,
dbo.#nResult3.EHZ, dbo.#nResult3.EXC,
dbo.#nResult3.FLG, dbo.#nResult3.IDQ, dbo.#nResult3.IFC,
dbo.#nResult3.INA, dbo.#nResult3.INF,
dbo.#nResult3.JDG,
dbo.#nResult3.KNW, dbo.#nResult3.LIM,
dbo.#nResult3.LGT, dbo.#nResult3.MIS,
dbo.#nResult3.MNT, dbo.#nResult3.OBS,
dbo.#nResult3.OPS,
dbo.#nResult3.PHY, dbo.#nResult3.PRB,
dbo.#nResult3.PRO, dbo.#nResult3.RES,
dbo.#nResult3.ROU, dbo.#nResult3.SKL,
dbo.#nResult3.TRG,
dbo.#nResult3.UNA, dbo.#nResult3.WXE,
dbo.#nResult2.CRW, dbo.#nResult2.WRK,
dbo.#nResult2.ENV, dbo.#nResult2.EQP,
dbo.#nResult2.ERR,
dbo.#nResult2.MED,
dbo.#nResult2.ORG, dbo.#nResult2.RDY,

```

```

dbo.#nResult2.SUP, dbo.#nResult2.VIO, dbo.#nResult1.MA,
dbo.#nResult1.MC,
dbo.#nResult1.MG,
dbo.#nResult1.WC
FROM dbo.#nResult3 INNER JOIN
dbo.#nResult2 ON dbo.#nResult3.Type_FK =
dbo.#nResult2.Type_FK INNER JOIN
dbo.#nResult1 ON dbo.#nResult3.Type_FK =
dbo.#nResult1.Type_FK

SELECT tblMishaps.Type_FK ,
Count(tblMishaps.MishapID) AS TotalMishaps INTO
#nResultTotal
FROM dbo.tblMishaps
GROUP BY tblMishaps.Type_FK

SELECT dbo.#nResultFinal.Type_FK ,
dbo.#nResultFinal.ADA, dbo.#nResultFinal.ASS,
dbo.#nResultFinal.ATT, dbo.#nResultFinal.COM,
dbo.#nResultFinal.CON, dbo.#nResultFinal.CRT,
dbo.#nResultFinal.DES,
dbo.#nResultFinal.DMG,
dbo.#nResultFinal.DOC, dbo.#nResultFinal.DUC,
dbo.#nResultFinal.EHZ, dbo.#nResultFinal.EXC,
dbo.#nResultFinal.FLG, dbo.#nResultFinal.IDQ,
dbo.#nResultFinal.IFC, dbo.#nResultFinal.INA,
dbo.#nResultFinal.INF,
dbo.#nResultFinal.JDG,
dbo.#nResultFinal.KNW, dbo.#nResultFinal.LIM,
dbo.#nResultFinal.LGT, dbo.#nResultFinal.MIS,
dbo.#nResultFinal.MNT, dbo.#nResultFinal.OBS,
dbo.#nResultFinal.OPS,
dbo.#nResultFinal.PHY, dbo.#nResultFinal.PRB,
dbo.#nResultFinal.PRO, dbo.#nResultFinal.RES,
dbo.#nResultFinal.ROU, dbo.#nResultFinal.SKL,
dbo.#nResultFinal.TRG,
dbo.#nResultFinal.UNA, dbo.#nResultFinal.WXE,
dbo.#nResultFinal.CRW, dbo.#nResultFinal.WRK,
dbo.#nResultFinal.ENV, dbo.#nResultFinal.EQP,
dbo.#nResultFinal.ERR,
dbo.#nResultFinal.MED,
dbo.#nResultFinal.ORG, dbo.#nResultFinal.RDY,
dbo.#nResultFinal.SUP, dbo.#nResultFinal.VIO,
dbo.#nResultFinal.MA, dbo.#nResultFinal.MC,
dbo.#nResultFinal.MG,
dbo.#nResultFinal.WC,
dbo.#nResultTotal.TotalMishaps,
dbo.tblMishapType.MishapTypeDefinition
FROM dbo.#nResultFinal INNER JOIN
dbo.#nResultTotal ON
dbo.#nResultFinal.Type_FK =dbo.#nResultTotal.Type_FK
INNER JOIN
dbo.tblMishapType ON
dbo.#nResultFinal.Type_FK =
dbo.tblMishapType.MishapTypeCode

```

```

DROP TABLE #nResultFinal
DROP TABLE #nResultTotal
DROP TABLE #nResult3
DROP TABLE #nResult2
DROP TABLE #nResult1

```

return

### 8-0-0-6-NelsonReportByYear

Alter Procedure [8-0-0-6-NelsonReportByYear]

As

SET NOCOUNT ON

```
CREATE TABLE #nResult3 (
    Year int,
    ADA int DEFAULT 0,
    ASS int DEFAULT 0,
    ATT int DEFAULT 0,
    COM int DEFAULT 0,
    CON int DEFAULT 0,
    CRT int DEFAULT 0,
    DES int DEFAULT 0,
    DMG int DEFAULT 0,
    DOC int DEFAULT 0,
    DUC int DEFAULT 0,
    EHZ int DEFAULT 0,
    EXC int DEFAULT 0,
    FLG int DEFAULT 0,
    IDQ int DEFAULT 0,
    IFC int DEFAULT 0,
    INA int DEFAULT 0,
    INF int DEFAULT 0,
    JDG int DEFAULT 0,
    KNW int DEFAULT 0,
    LGT int DEFAULT 0,
    LIM int DEFAULT 0,
    MIS int DEFAULT 0,
    MNT int DEFAULT 0,
    OBS int DEFAULT 0,
    OPS int DEFAULT 0,
    PHY int DEFAULT 0,
    PRB int DEFAULT 0,
    PRO int DEFAULT 0,
    RES int DEFAULT 0,
    ROU int DEFAULT 0,
    SKL int DEFAULT 0,
    TRG int DEFAULT 0,
    UNA int DEFAULT 0,
    UNK int DEFAULT 0,
    WXE int DEFAULT 0
)
```

```
CREATE TABLE #nResult2 (
    Year int,
    CRW int DEFAULT 0,
    ENV int DEFAULT 0,
    EQP int DEFAULT 0,
    ERR int DEFAULT 0,
    MED int DEFAULT 0,
    ORG int DEFAULT 0,
    RDY int DEFAULT 0,
    SUP int DEFAULT 0,
    UNK int DEFAULT 0,
    VIO int DEFAULT 0,
    WRK int DEFAULT 0
)
```

```
CREATE TABLE #nResult1 (
    Year int,
    MA int DEFAULT 0,
    MC int DEFAULT 0,
    MG int DEFAULT 0,
    UN int DEFAULT 0,
    WC int DEFAULT 0
)
```

```
)
CREATE TABLE #nResultFinal (
    Year int,
    ADA int DEFAULT 0,
    ASS int DEFAULT 0,
    ATT int DEFAULT 0,
    COM int DEFAULT 0,
    CON int DEFAULT 0,
    CRT int DEFAULT 0,
    DES int DEFAULT 0,
    DMG int DEFAULT 0,
    DOC int DEFAULT 0,
    DUC int DEFAULT 0,
    EHZ int DEFAULT 0,
    EXC int DEFAULT 0,
    FLG int DEFAULT 0,
    IDQ int DEFAULT 0,
    IFC int DEFAULT 0,
    INA int DEFAULT 0,
    INF int DEFAULT 0,
    JDG int DEFAULT 0,
    KNW int DEFAULT 0,
    LIM int DEFAULT 0,
    LGT int DEFAULT 0,
    MIS int DEFAULT 0,
    MNT int DEFAULT 0,
    OBS int DEFAULT 0,
    OPS int DEFAULT 0,
    PHY int DEFAULT 0,
    PRB int DEFAULT 0,
    PRO int DEFAULT 0,
    RES int DEFAULT 0,
    ROU int DEFAULT 0,
    SKL int DEFAULT 0,
    TRG int DEFAULT 0,
    UNA int DEFAULT 0,
    WXE int DEFAULT 0,
    CRW int DEFAULT 0,
    WRK int DEFAULT 0,
    ENV int DEFAULT 0,
    EQP int DEFAULT 0,
    ERR int DEFAULT 0,
    MED int DEFAULT 0,
    ORG int DEFAULT 0,
    RDY int DEFAULT 0,
    SUP int DEFAULT 0,
    VIO int DEFAULT 0,
    MA int DEFAULT 0,
    MC int DEFAULT 0,
    MG int DEFAULT 0,
    WC int DEFAULT 0
)
```

-----FOR THIRD LEVEL FACTORS

--Build a temp table and update the null values to 'None'

```
SELECT MishapID, [3rdLevelCode], Year INTO
#nTemp3
```

```
FROM [vwFlanReports-2-7-Year3]
```

```
UPDATE #nTemp3
```

```
SET Year = '0'
```

```
WHERE Year is null
```

```
--Now run the crosstab
```

```
INSERT #nResult3
```

```

EXEC dbo.rac @grpcol='Year',
@pvtcol='[3rdLevelCode]', @transform='count(*)', @from
=#nTemp3', @where=',
    @printagg='n',@grand_totals='n',
@row_totals='n', @emptycell='0'

----- FOR SECOND LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [2ndLevelCode], Year INTO
#nTemp2
FROM [vwFlanReports-2-7-Year2]

UPDATE #nTemp2
SET Year = '0'
WHERE Year is null
--Now run the crosstab
INSERT #nResult2
EXEC dbo.rac @grpcol='Year',
@pvtcol='[2ndLevelCode]', @transform='count(*)', @from
=#nTemp2', @where=',
    @printagg='n',@grand_totals='n',
@row_totals='n', @emptycell='0'

----- FOR FIRST LEVEL FACTORS
--Build a temp table and update the null values to 'None'
SELECT MishapID, [1stLevelCode], Year INTO
#nTemp1
FROM [vwFlanReports-2-7-Year1]

UPDATE #nTemp1
SET Year = '0'
WHERE Year is null
--Now run the crosstab
INSERT #nResult1
EXEC dbo.rac @grpcol='Year',
@pvtcol='[1stLevelCode]', @transform='count(*)', @from
=#nTemp1', @where=',
    @printagg='n',@grand_totals='n',
@row_totals='n', @emptycell='0'

-----

INSERT #nResultFinal
SELECT dbo.#nResult3.Year,
dbo.#nResult3.ADA, dbo.#nResult3.ASS,
dbo.#nResult3.ATT, dbo.#nResult3.COM,
dbo.#nResult3.CON, dbo.#nResult3.CRT,
dbo.#nResult3.DES,
    dbo.#nResult3.DMG,
dbo.#nResult3.DOC, dbo.#nResult3.DUC,
dbo.#nResult3.EHZ, dbo.#nResult3.EXC,
dbo.#nResult3.FLG, dbo.#nResult3.IDQ, dbo.#nResult3.IFC,
dbo.#nResult3.INA, dbo.#nResult3.INF,
    dbo.#nResult3.JDG,
dbo.#nResult3.KNW, dbo.#nResult3.LIM,
dbo.#nResult3.LGT, dbo.#nResult3.MIS,
dbo.#nResult3.MNT, dbo.#nResult3.OBS,
    dbo.#nResult3.OPS,
dbo.#nResult3.PHY, dbo.#nResult3.PRB,
dbo.#nResult3.PRO, dbo.#nResult3.RES,
dbo.#nResult3.ROU, dbo.#nResult3.SKL,
    dbo.#nResult3.TRG,
dbo.#nResult3.UNA, dbo.#nResult3.WXE,
dbo.#nResult2.CRW, dbo.#nResult2.WRK,
dbo.#nResult2.ENV, dbo.#nResult2.EQP,
dbo.#nResult2.ERR,

```

```

    dbo.#nResult2.MED,
dbo.#nResult2.ORG, dbo.#nResult2.RDY,
dbo.#nResult2.SUP, dbo.#nResult2.VIO, dbo.#nResult1.MA,
dbo.#nResult1.MC,
    dbo.#nResult1.MG,
dbo.#nResult1.WC
FROM dbo.#nResult3 INNER JOIN
    dbo.#nResult2 ON dbo.#nResult3.Year =
dbo.#nResult2.Year INNER JOIN
    dbo.#nResult1 ON dbo.#nResult3.Year =
dbo.#nResult1.Year

SELECT #nTemp3.Year, Count(Distinct
#nTemp3.MishapID) AS TotalMishaps INTO #nResultTotal
From #nTemp3
Group By #nTemp3.Year

SELECT dbo.#nResultFinal.Year ,
dbo.#nResultFinal.ADA, dbo.#nResultFinal.ASS,
dbo.#nResultFinal.ATT, dbo.#nResultFinal.COM,
dbo.#nResultFinal.CON, dbo.#nResultFinal.CRT,
dbo.#nResultFinal.DES,
    dbo.#nResultFinal.DMG,
dbo.#nResultFinal.DOC, dbo.#nResultFinal.DUC,
dbo.#nResultFinal.EHZ, dbo.#nResultFinal.EXC,
dbo.#nResultFinal.FLG, dbo.#nResultFinal.IDQ,
dbo.#nResultFinal.IFC, dbo.#nResultFinal.INA,
dbo.#nResultFinal.INF,
    dbo.#nResultFinal.JDG,
dbo.#nResultFinal.KNW, dbo.#nResultFinal.LIM,
dbo.#nResultFinal.LGT, dbo.#nResultFinal.MIS,
dbo.#nResultFinal.MNT, dbo.#nResultFinal.OBS,
    dbo.#nResultFinal.OPS,
dbo.#nResultFinal.PHY, dbo.#nResultFinal.PRB,
dbo.#nResultFinal.PRO, dbo.#nResultFinal.RES,
dbo.#nResultFinal.ROU, dbo.#nResultFinal.SKL,
    dbo.#nResultFinal.TRG,
dbo.#nResultFinal.UNA, dbo.#nResultFinal.WXE,
dbo.#nResultFinal.CRW, dbo.#nResultFinal.WRK,
dbo.#nResultFinal.ENV, dbo.#nResultFinal.EQP,
dbo.#nResultFinal.ERR,
    dbo.#nResultFinal.MED,
dbo.#nResultFinal.ORG, dbo.#nResultFinal.RDY,
dbo.#nResultFinal.SUP, dbo.#nResultFinal.VIO,
dbo.#nResultFinal.MA, dbo.#nResultFinal.MC,
    dbo.#nResultFinal.MG,
dbo.#nResultFinal.WC,
    dbo.#nResultTotal.TotalMishaps
FROM dbo.#nResultFinal INNER JOIN
    dbo.#nResultTotal ON
dbl.#nResultFinal.Year = dbo.#nResultTotal.Year

DROP TABLE #nResultFinal
DROP TABLE #nResultTotal
DROP TABLE #nResult3
DROP TABLE #nResult2
DROP TABLE #nResult1

```

return

### **8-0-0-9-NelsonCronoMishaps**

Alter Procedure [8-0-0-9-NelsonCronoMishaps]

As

```
SELECT      tblMishaps.MishapDate,      tblMishaps.Aircraft_FK,      tblMishaps.Class_FK,  
            tblMishaps.Type_FK, tblMishaps.OrgID_FK, tblMishaps.ShortDescription
```

```
FROM tblMishaps
```

```
ORDER BY tblMishaps.MishapDate
```

```
return
```



### **9-0-0-1-Lookups Without ALL**

Alter Procedure [9-0-0-1 -flanLookupAircraft]

As  
set nocount on

SELECT \*FROM dbo.tblAircraft  
ORDER BY AircraftTypeModel

Return

---

Alter Procedure [9-0-0-1 -flanLookupClass]

As  
set nocount on

SELECT \*FROM dbo.tblMishapClass  
ORDER BY MishapClassCode

Return

---

Alter Procedure [9-0-0-1 -flanLookupDBType]

As  
set nocount on

SELECT DatabaseType  
FROM dbo.tblDatabaseType  
WHERE DatabaseType <> 'O'

Return

---

Alter Procedure [9-0-0-1 -flanLookupFactors]

As  
set nocount on

SELECT \*FROM dbo.tblFactors  
ORDER BY [3rdLevelDesc]

Return

---

Alter Procedure [9-0-0-1 -flanLookupLocation]

(  
          @DatabaseType    varchar(1) = "M"  
)

As

set nocount on

SELECT dbo.tblMishapLocation.MishapLocationID,  
        dbo.tblMishapLocation.MishapLocation

FROM dbo.tblDatabaseType INNER JOIN  
      dbo.tblMishapLocation ON  
      dbo.tblDatabaseType.DatabaseType =  
      dbo.tblMishapLocation.DatabaseType

WHERE  
      dbo.tblMishapLocation.DatabaseType=dbo.tblDatabaseType.  
      DatabaseType

ORDER BY dbo.tblMishapLocation.MishapLocation

return

---

Alter Procedure [9-0-0-1 -flanLookupOrganization]

As  
set nocount on

SELECT dbo.tblOrganization.OrgID,  
        dbo.tblOrganization.OrgName

FROM dbo.tblOrganization INNER JOIN  
      dbo.tblDatabaseType ON dbo.tblOrganization.DatabaseType  
      = dbo.tblDatabaseType.DatabaseType

WHERE  
      dbo.tblOrganization.DatabaseType=dbo.tblDatabaseType.Da  
      tabaseType

ORDER BY OrgID

return

---

Alter Procedure [9-0-0-1 -flanLookupType]

As  
set nocount on

SELECT \*FROM dbo.tblMishapType  
ORDER BY MishapTypeCode

Return

---

## **9-0-0-2-LookupsWithALL**

Alter Procedure [9-0-0-2-flanLookupAircraftAll]

As

set nocount on

SELECT AircraftTypeModel, AircraftCategory,  
AircraftDescription FROM dbo.tblAircraft

UNION

Select '<All>' as AllChoice, '<All>' as AllChoice2, '<All>' as  
AllChoice3 FROM tblAircraft

ORDER BY AircraftTypeModel

return

---

Alter Procedure [9-0-0-2-flanLookupClassAll]

As

set nocount on

SELECT MishapClassCode, MishapClassDefinition FROM  
dbo.tblMishapClass

UNION

Select '<All>' as AllChoice, '<All>' as AllChoice2 FROM  
dbo.tblMishapClass

ORDER BY MishapClassCode

Return

---

Alter Procedure [9-0-0-2-flanLookupFactorsAll1Level]

As

set nocount on

SELECT DISTINCT [1stLevelCode], [1stLevelDesc]  
FROM dbo.tblFactors

UNION

Select '<All>' as AllChoice, '<All>' as AllChoice2 FROM  
dbo.tblFactors

ORDER BY [1stLevelDesc]

return

---

Alter Procedure [9-0-0-2-flanLookupFactorsAll2Level]

As

set nocount on

SELECT DISTINCT [2ndLevelCode], [2ndLevelDesc]  
FROM dbo.tblFactors

UNION

---

Select '<All>' as AllChoice, '<All>' as AllChoice2 FROM  
dbo.tblFactors

ORDER BY [2ndLevelDesc]

Return

---

Alter Procedure [9-0-0-2-flanLookupFactorsAll3Level]

As

set nocount on

SELECT DISTINCT [3rdLevelCode], [3rdLevelDesc]  
FROM dbo.tblFactors

UNION

Select '<All>' as AllChoice, '<All>' as AllChoice2 FROM  
dbo.tblFactors

ORDER BY [3rdLevelDesc]

Return

---

Alter Procedure [9-0-0-2-flanLookupLocationAll]

(  
                    @DatabaseType      varchar(1) = "M"  
)

As

set nocount on

SELECT dbo.tblMishapLocation.MishapLocationID,  
       dbo.tblMishapLocation.MishapLocation

FROM  dbo.tblMishapLocation INNER JOIN  
       dbo.tblDatabaseType ON  
       dbo.tblMishapLocation.DatabaseType =  
       dbo.tblDatabaseType.DatabaseType

WHERE  
       dbo.tblMishapLocation.DatabaseType=dbo.tblDatabaseType.  
       DatabaseType

UNION

Select '<All>' as AllChoice, '<All>' as AllChoice2 FROM  
dbo.tblMishapLocation

ORDER BY dbo.tblMishapLocation.MishapLocation

Return

---

Alter Procedure [9-0-0-2-flanLookupOrganizationAll]

As

set nocount on

SELECT  dbo.tblOrganization.OrgID,  
          dbo.tblOrganization.OrgName

```
FROM dbo.tblOrganization INNER JOIN  
dbo.tblDatabaseType ON dbo.tblOrganization.DatabaseType  
= dbo.tblDatabaseType.DatabaseType
```

```
WHERE  
dbo.tblOrganization.DatabaseType=dbo.tblDatabaseType.Da  
tabaseType
```

```
UNION
```

```
Select '<All>' as AllChoice, '<All>' as AllChoice2 FROM  
dbo.tblOrganization
```

```
ORDER BY OrgID
```

```
return
```

---

```
Alter Procedure [9-0-0-2-flanLookupTypeAll]
```

```
As  
set nocount on
```

---

```
SELECT MishapTypeCode, MishapTypeDefinition FROM  
dbo.tblMishapType
```

```
UNION
```

```
Select '<All>' as AllChoice, '<All>' as AllChoice2 FROM  
dbo.tblMishapType
```

```
ORDER BY MishapTypeCode
```

```
return
```

---

```
Alter Procedure [9-0-0-2-flanModifiedLookupYear]
```

```
As  
set nocount on
```

```
SELECT DISTINCT  
DatePart("yyyy",[tblMishaps].[MishapDate]) AS Expr1  
FROM dbo.tblMishaps;
```

```
return
```

---

## **RAC**

RAC is an application that runs on SQL Server and produces 2 dimensional cross-tab reports. It was designed by Steve Dassin and was included in HFACS with his permission [Ref. 31].

RAC has various options that make it possible to enhance the traditional Access-JET cross-tab functionality by providing additional capabilities over those in Access. RAC has a number of report like format capabilities that enhance the appearance of table data. In addition to producing cross-tab reports, RAC can be used to transpose fields, split delimited strings and create delimited strings. RAC is written in transact-SQL exclusively for SQL Server version 7.0 and above. A set oriented approach is employed in most places and RAC does NOT use any cursors.

## LIST OF REFERENCES

1. Schmidt, J. & Lawson, D., Aviation Maintenance Human Factors Accident Analysis. Power Point Presentation. Adapted form Reason's Swiss Cheese Model. Monterey, CA: School of Aviation Safety, 2000.
2. Wood, B. P., Information Management System development for the Characterization and Analysis of Human Error in Naval Aviation Maintenance Related Mishaps, 2000.
3. N00S Aviation Safety Home Page, Operational & Risk Management, [http://www.navres.navy.mil/navresfor/navair/safety/av\\_saftey.html](http://www.navres.navy.mil/navresfor/navair/safety/av_saftey.html).
4. United States Navy Aviation Safety Center Home Page, Aviation Directorate HFACS-ME, <http://www.safetycenter.navy.mil/aviation/Presentations/qhfamm6/sld005.htm>.
5. Booch, Rumbaugh, & Jacobson M., *The Unified Modeling Language User Guide*. Addison Wesley Longman Inc., Reading, MASS, 1997.
6. Kent Beck & Ward Cunningham, *OOPSLA'89 Conference Proceedings*, October 1-6, New Orleans, Louisiana, 1989.
7. Muller, Robert J., Database Design for Smarties Using UML for Data Modeling. Morgan, Kaufmann Publishers, San Francisco, CA., 1999.
8. Blackburn, Ian, Professional Access 2000 Programming. Wrox Press Ltd., Birmingham, UK, 2000.
9. Doyle, Casey D., Microsoft Office 97 Visual Basic Programmer's Guide. Microsoft Press, Redmond, Washington, 1997.
10. Doyle, Casey D., Microsoft Office 97 Resource Kit. Microsoft Press, Redmond, Washington, 1997.
11. Halvorson, Michael, Step-by-Step Microsoft Visual Basic 6.0. Microsoft Press, Redmond, Washington, 1998.
12. Prague & Irwin, Microsoft Access 2000 Bible. IDG Books Worldwide, Inc., Foster City, CA., 1999.
13. Solomon, Christine, Microsoft Office 97 Developer's Handbook. Microsoft Press, Redmond, Washington, 1997.

14. Williams, Charles, Professional Visual Basic 6 Databases, Wrox Press, Birmingham, UK., 1999.
15. Universal Data Access Web Site, ActiveX Direct Objects,  
<http://www.microsoft.com/data/ado/default.htm>
16. Microsoft Web Site, Upgrading to Access 2002,  
<http://www.microsoft.com/Office/ORK/xp/WELCOME/depf05.htm>.
17. Microsoft Developer Network Web Site, Upgrading Visual Basic 6.0 Applications To Visual Basic.NET,  
<http://msdn.microsoft.com/vstudio/nextgen/technology/vbupgrade.asp>.
18. Microsoft Developer Network Web Site, Preparing Your Visual Basic 6.0 Applications for the Upgrade to Visual Basic.NET,  
<http://msdn.microsoft.com/library/default.asp?URL=/library/techart/vb6tovbdotnet.htm>.
19. Boehm, Barry, *Software Risk Management*, IEEE Computer Society Press, 1989.
20. Microsoft Product Support Services Web Site, Frequently Asked Question - SQL Server 2000 - Upgrade,  
<http://support.microsoft.com/support/kb/articles/Q261/3/34.ASP>.
21. Microsoft MSDN Online Magazine Web Site, SQL Server and DMO: Distributed Management Objects Enable Easy Task Automation,  
<http://msdn.microsoft.com/msdnmag/issues/01/05/sqlldmo/sqlldmo.asp>.
22. Microsoft Product Support Services Web Site, Incompatibility Issues Between Access 2000 Projects and SQL Server 2000,  
<http://support.microsoft.com/support/kb/articles/Q269/8/24.ASP>.
23. Microsoft Office Web Site, Access 2000 and SQL Server 2000 Readiness Update,  
<http://office.microsoft.com/downloads/2000/Accsql.aspx>.
24. Microsoft MSDN Online Library Web Site, Distributing Custom Icons with Your Microsoft Office 2000 Applications,  
<http://msdn.microsoft.com/library/default.asp>.
25. Shappell, S. & Wiegmann, D., *A Human Factors Analysis of Post-Accident Data: Applying Theoretical Taxonomies of Human Error and A Human Error Approach to Accident Investigation: The Taxonomy of Unsafe Operations*, The International Journal of Aviation Psychology, 7, (4), 67-81 & 269-291, 1997.

26. Schmidt, J., Schmorrow, D., & Hardee, M. A., *Preliminary Human Factors Analysis of Naval Aviation Maintenance Related Mishaps* (983111), Society of Automotive Engineers, Inc., 1997.
27. Reason, J., *Human Error*. Cambridge, UK: Cambridge Press, 1990.
28. Heinrich, H., *Industrial Accident Prevention*, 4th ed. New York, NY: McGraw-Hill, 1959.
29. Edwards, E., *Introductory Overview from Human Factors in Aviation*, (Weiner, E. L. & Nagel, D.C., Eds.) San Diego, CA: Academic Press. 3-25, 1988.
30. Raghu Ramkrishnan & Johannes Gehrke, *Database Management Systems*. McGraw-Hill Companies Inc., Boston, MA., 2000.
31. Replacement For Access Crosstab Website, Steve Dassin,  
<http://www.angelfire.com/ny4/rac/>.
32. Schmidt, J. (1998). *Human Factors Accident Classification System Analysis of Selected National Transportation Safety Board Maintenance Related Mishaps, Chapter 8*. Unpublished Manuscript.
33. Schmorrow D. *A Human Error and Analysis Model of Naval Aviation Maintenance Related Mishaps*, Master's Thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA (1998).
34. Fry, A.D. *Modeling and Analysis of Human Error in Naval Aviation Maintenance Mishaps*, Master's Thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA (2000).

THIS PAGE INTENTIONALLY LEFT BLANK



## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center  
Ft. Belvoir, Virginia
2. Dudley Knox Library  
Naval Postgraduate School  
Monterey, California
3. CAPT John K. Schmidt (NAVY)  
Naval Safety Center  
Norfolk, Virginia  
brainsqzer@aol.com
4. CAPT(R) George Zolla  
Naval Postgraduate School  
Monterey, California  
gazolla@nps.navy.mil
5. Professor Thomas Wu  
Naval Postgraduate School  
Monterey, California  
ctwu@nps.navy.mil
6. LtCDR Chris Eagle (NAVY)  
Naval Postgraduate School  
Monterey, California  
cseagle@cs.nps.navy.mil
7. MAJ Thomas P. Flanders (ARMY)  
Naval Postgraduate School  
Monterey, California  
tpflande@nps.navy.mil
8. MAJ Scott K. Tufts (ARMY)  
Naval Postgraduate School  
Monterey, California  
sktufts@nps.navy.mil

9. Chairman, Computer Science Department  
Naval Postgraduate School  
Monterey, California  
cschair@nps.navy.mil